UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Identificación de terminales mediante su tráfico WiFi

Máster Universitario en Ingeniería de Telecomunicación

Autor: BLÁZQUEZ SÁNCHEZ, Alberto
Tutor: GARCÍA DORADO, José Luis

FECHA: Septiembre, 2017

IDENTIFICACIÓN DE TERMINALES MÓVILES MEDIANTE SU TRÁFICO WIFI

AUTOR: Alberto Blázquez Sánchez
TUTOR: José Luis García Dorado

Máster Universitario en Ingeniería de Telecomunicación
Dpto. Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Septiembre de 2017

Resumen

A medida que el uso de terminales móviles ha aumentado, los operadores de telefonía se han visto en la obligación de dotar de aún más inteligencia a sus redes para poder emplear sus recursos y proporcionar sus servicios de un modo eficiente. Actualmente, disponen de un gran volumen de información sobre sus usuarios y, con las herramientas adecuadas, podrían llegar a utilizarla para, por ejemplo, ofrecer un servicio personalizado ante problemas técnicos, realizar ofertas adecuadas a la capacidad de los terminales de sus clientes, reservar mayores capacidades de ancho de banda durante periodos específicos de tiempo o determinar perfiles de usuarios según su capacidad adquisitiva. Para ello, necesitan contar con la mayor cantidad de información posible para poder hacer predicciones sobre el tipo de contenidos que se demandarán y la cantidad de recursos que requerirán.

Sin embargo, cuando los dispositivos móviles se encuentran conectados a un punto de acceso de red inalámbrica a través de la tecnología Wifi, el tráfico generado no se intercambia de manera directa entre los terminales y las estaciones base, sino que se realiza a través de un dispositivo intermedio (p. ej., a través de un router), perdiendo así el control de la actividad que genera cada uno de ellos de manera individual.

La motivación principal detrás de este Trabajo es dotar de mayor información a los operadores para que puedan adaptar sus servicios y ofertas y proporcionar controles de calidad avanzados. En concreto, se propone la implementación de un mecanismo que permita a los operadores conocer qué dispositivos se encuentran conectados a la típica red Wifi doméstica, informando sobre el fabricante y modelo de todos ellos, además de otros datos relevantes, como las páginas que se encuentran visitando. De esta forma, dispondrán de información relativa a los hábitos de diferentes grupos de usuarios, lo que les permitirá entender la carga de tráfico en todos los puntos de la red y buscar tendencias para detectar cambios en sus prácticas de uso.

Palabras clave

WIFI, TERMINALES MÓVILES, IDENTIFICACIÓN, APLICACIÓN, TRÁFICO, HTTP.

Abstract

As the use of mobile terminals has increased, telephony operators have been forced to provide even more intelligence to their networks in order to use their resources and provide services efficiently. Currently, they have a very large amount of information about their users and, with the right tools, they could exploit it to, for example, offer a personalized service to technical problems, make offers appropriate to the capacity of the terminals of their clients, reserve greater bandwidth capacity during specific periods of time or determine user profiles according to their purchasing power. To do this, they need to have as much information as possible to be able to make predictions about the type of content that will be demanded and the amount of resources that will be required.

However, when mobile devices are connected to a wireless network access point via Wifi technology, the generated traffic is not exchanged directly between the terminals and the base stations, but is done through an intermediate device (e.g., through a router), thus losing control of the activity generated by each of them individually.

The main motivation behind this work is to provide more information to operators so that they can adapt their services and offerings and provide advanced quality controls. Specifically, it is proposed the implementation of a mechanism that allows operators to know which devices are connected to the typical domestic Wifi network, informing about the manufacturer and model of all of them, as well as other relevant data, such as the visited pages. In this way, they will have information about the habits of different groups of users, which will allow them to understand the traffic load at all points of the network and look for trends to detect changes in their usage practices.

Keywords

WIFI, MOBILE TERMINALS, IDENTIFICATION, APPLICATION, TRAFFIC, HTTP.

Agradecimientos

Quiero expresar mi agradecimiento al Departamento de Tecnología Electrónica y de las Comunicaciones, y especialmente a José Luis, mi tutor de este TFM, por su incondicional ayuda, colaboración y disponibilidad. Los resultados de este Trabajo son en gran parte gracias a su dedicación. Ha sido un placer trabajar en su mismo equipo durante estos meses.

También quiero hacer una mención especial a Javier Ramos por su ayuda desinteresada para la instalación del prototipo, sin la cual no habríamos podido validar el funcionamiento en un entorno real.

Y, por último, deseo agradecerle a Cris, a mi familia y a mis amigos el apoyo que me han dado a lo largo de toda esta etapa. Gracias por vuestro tiempo y paciencia.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 Introducción

1.1 Motivación

El uso de smartphones ha crecido de manera notable en los últimos años, lo que ha llevado a los operadores de telefonía a dotar de aún más inteligencia a sus redes para poder emplear sus recursos y proporcionar sus servicios de un modo eficiente. El creciente uso de terminales móviles ha supuesto que el número de usuarios que deben soportar las redes haya crecido de forma considerable y, por tanto, se haya producido un notable aumento de la densidad de tráfico. Este hecho, unido a la progresiva necesidad de velocidades de transmisión superiores y a una agresiva competencia entre operadores, se ha traducido en la aparición de diferentes mecanismos de gestión que, a día de hoy, resultan imprescindibles para proveer la calidad de servicio a los usuarios que éstos exigen.

Por un lado, en términos de satisfacción del usuario, se tiene como objetivo dar un servicio personalizado ante problemas técnicos. Además, los operadores pueden iniciar procesos de monetización de la información que poseen de sus usuarios, por ejemplo, mediante campañas promocionales o de marketing personalizadas por usuario e, incluso, comercializar esta fuente de información de forma equivalente a lo que hace *Google Ads* [1].

Para los operadores, puede resultar especialmente relevante el potencial conocimiento sobre los dispositivos conectados a su red. Un análisis particular sobre cada usuario les permitiría proporcionar a sus clientes ofertas adecuadas a la capacidad de sus teléfonos móviles, proponiéndoles promociones para la actualización de sus terminales o dándoles una atención muy concreta y detallada en el caso de que éstos tuviesen problemas de configuración, generando, incluso, procesos semi-automáticos. De la misma forma, se podrían determinar perfiles de usuarios según su capacidad adquisitiva.

Por otro lado, en términos de rendimiento, si se detecta, por ejemplo, que muchos de los dispositivos de una zona determinada tienen un tamaño de pantalla grande, se puede intuir que éstos probablemente consumirán una cantidad alta de contenido de vídeo en streaming, lo que requiere de un elevado ancho de banda. Del mismo modo, si la mayoría de dispositivos son *smartphones* de gama media-baja y, además, su Sistema Operativo no es razonablemente potente, el operador podrá concluir que uno de sus usos más probables es el intercambio de información a través de redes sociales, lo cual requiere de actualizaciones y cargas muy numerosas y de pequeño tamaño [2]. Esta primera aproximación permite deducir que es necesario tener un conocimiento exacto de los elementos que componen la red para poder cubrir sus necesidades particulares.

Además, los datos de inteligencia de red también pueden permitir identificar celdas específicas en las que se utilizan con gran frecuencia aplicaciones que necesitan muchos recursos. Por ejemplo, si el uso de video chats móviles es muy habitual en el centro de la ciudad durante el horario laboral, el operador puede aumentar el ancho de banda disponible para la célula responsable de esa área durante dicho período de tiempo.

Actualmente, según un estudio de GSMA (GSM Association, 2017), el número de usuarios móviles a nivel mundial supera los 5.000 millones, y la previsión para el año 2020 es que

se superen los 5.700 millones [3]. El estudio "Informe Mobile en España y en el Mundo 2017" [4], también publicado por GSMA, revela que nuestro país lidera el ranking mundial en utilización de teléfonos móviles, contando con un 88% sobre el total de la población (Figura 1-1). Estas cifras suponen un importante reto para los operadores de telefonía, puesto que el número de servicios que consume cada usuario a través de sus redes es cada vez mayor y, al mismo tiempo, sus redes deben soportar un número de usuarios cada vez más elevado.

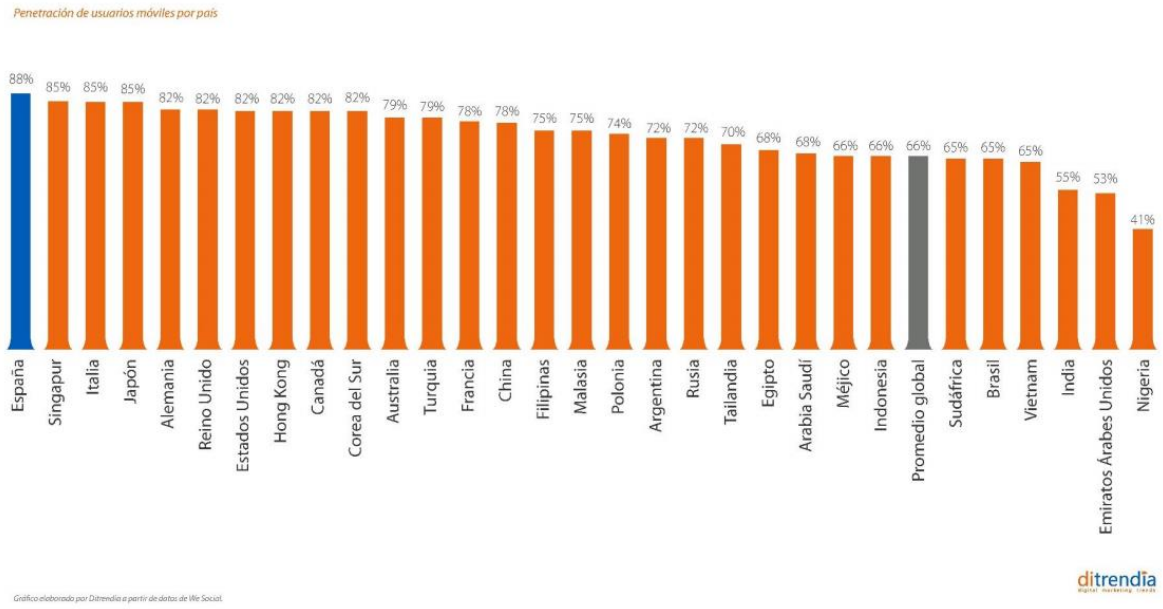

Figura 1-1. Penetración de usuarios móviles por país.

(Fuente: *https://ditrendia.es/informe-mobile-espana-mundo-2017*).

Los operadores deben tener la capacidad de adaptarse a las transmisiones de datos, etiquetando diferentes tipos de tráfico y asignando distintos anchos de banda en función de las necesidades de las aplicaciones que se encuentran ejecutando sus usuarios. Por tanto, necesitan disponer de la mayor cantidad de información posible sobre la red y sus clientes para poder hacer predicciones sobre el tipo de contenidos que se demandarán y el volumen de recursos que requerirán.

En definitiva, el control sobre el número y tipo de terminales de una red da a las compañías proveedoras de servicios de Internet la posibilidad de obtener una mejor descripción de los tipos de usuarios de los que disponen, pudiendo definir así de una forma bastante exacta el perfil de los clientes a monetizar.

Sin embargo, cuando los dispositivos se encuentran conectados a un punto de acceso de red inalámbrica a través de la tecnología Wifi, el tráfico web generado no se intercambia de manera directa entre los terminales y las estaciones base, sino que se realiza a través de un dispositivo intermedio (p. ej., a través de un router), perdiendo así el control de la actividad que genera cada uno de ellos de manera individual.

Por tanto, la motivación principal detrás de este Trabajo es proporcionar mayor información a los operadores para que puedan adaptar sus servicios y ofertas y

proporcionar controles de calidad avanzados. En concreto, se propone la implementación de un mecanismo que permita a los operadores conocer qué dispositivos se encuentran conectados a la típica red Wifi doméstica, informando sobre el fabricante y modelo de todos ellos, además de otros datos relevantes, como las páginas que se encuentran visitando. De esta forma, dispondrán de información relativa a los hábitos de diferentes grupos de usuarios, lo que les permitirá entender la carga de tráfico en todos los puntos de la red y buscar tendencias para detectar cambios en sus prácticas de uso.

1.2 Objetivos

La finalidad principal de este Trabajo Fin de Master ha sido desarrollar una herramienta que permita detectar, con la mayor fiabilidad posible, qué dispositivos se encuentran conectados a una red Wifi. Para ello, los objetivos marcados inicialmente fueron los siguientes:

- Profundizar en el conocimiento sobre el estado actual de las redes de telefonía móvil y las distintas técnicas empleadas para gestionar el tráfico que soportan, con el fin de entender cuál es la motivación que hay detrás de este Trabajo, qué necesidades de los operadores se pretenden cubrir con él y definir adecuadamente los servicios a desarrollar.

- Investigar sobre el estado del arte en el ámbito de estudio de este TFM y analizar las técnicas existentes para la identificación de modelos de dispositivos móviles a partir de su tráfico, tratando de concretar cuál de las propuestas desarrolladas en trabajos similares se ajusta a nuestro caso de estudio.

- Conocer y examinar las características concretas del tráfico que generan los *smartphones* según su Sistema Operativo, fabricante y modelo para identificar algún tipo de pauta que permita determinar qué dispositivos se están utilizando en una red.

- Aprender a utilizar herramientas de captura de tráfico en tiempo real, necesarias para poder controlar y analizar los paquetes capturados durante la ejecución de la aplicación.

- Realizar distintos análisis mediante simulaciones en un entorno de pruebas, emulando la utilización de dispositivos móviles reales desde un navegador web que imite el funcionamiento de terminales existentes.

- Conocer cuál es el proceso de instalación que se debe llevar a cabo para lanzar una aplicación dada en el router de una red concreta.

- Trasladar las herramientas desarrolladas, potencialmente un prototipo comercial, a un entorno real y validar su funcionamiento a partir de un escenario conocido, estudiando de forma detallada las características concretas de la aplicación, valorando su efectividad y extrayendo las conclusiones oportunas.

1.3 Organización de la memoria

En el Capítulo 2 del documento se muestra una visión global del estado del arte en el ámbito relacionado con este proyecto. En él se realiza, de forma general, una investigación documental sobre la información que manejan los operadores móviles y las posibilidades que existen para la detección de terminales dentro de una red. Más tarde, se profundiza en cuestiones relacionadas con esta materia, explicando con detalle las técnicas y protocolos que se han empleado en este Trabajo. Además, se lleva a cabo un estudio sobre otros trabajos relacionados con la propuesta desarrollada, comparando las metodologías empleadas en ellos con la planteada para este proyecto.

A continuación, el Capítulo 3 describe las funcionalidades que ofrece la aplicación desarrollada y explica su funcionamiento a alto nivel. En este apartado se expone cuáles son los conceptos sobre los que se basa la aplicación y cómo han sido trasladados a la misma.

El Capítulo 4 define la metodología empleada en la realización de este Trabajo Fin de Máster. En él, se detallan cuáles han sido las herramientas empleadas y otras cuestiones específicas, como la documentación utilizada y las distintas fases necesarias para el desarrollo. Además, se exponen algunas de las cuestiones particulares de mayor relevancia de la aplicación, como, por ejemplo, las herramientas utilizadas para el control de versiones o para la captura de tráfico.

El Capítulo 5 se centra en aspectos relacionados con la implementación del código de la aplicación. En el primer apartado de este capítulo se exponen los pasos seguidos durante el desarrollo. A continuación, se ofrece una descripción de los ficheros de los que se compone y, por último, se detallan algunos de los requisitos básicos necesarios para que el programa funcione adecuadamente.

Por su parte, el Capítulo 6 expone la parte relacionada con la puesta en marcha de la aplicación sobre el router empleado para las pruebas. En primer lugar, se explica en qué consistió el proceso de instalación y qué consideraciones fueron tenidas en cuenta para configurar el router y, a continuación, se detalla cuáles fueron los resultados obtenidos tras la implantación del programa.

En el siguiente apartado de la Memoria, correspondiente al Capítulo 7, se recogen las conclusiones más relevantes obtenidas durante la realización del Trabajo Fin de Máster, así como algunas posibles líneas futuras a seguir tras la conclusión del mismo.

Seguidamente, se muestra la bibliografía utilizada para el desarrollo del Trabajo y documentación de interés relacionada con los temas abordados en la memoria, así como un Glosario con términos y expresiones empleados en el texto, incluyendo sus siglas junto a su significado.

Por último, se ha incluido un Anexo en el que se detallan las distintas cabeceras existentes dentro del protocolo HTTP, analizadas como parte del tráfico generado por los dispositivos durante la ejecución del programa.

2 Estado del arte

2.1 Introducción

Los operadores móviles tienen acceso a todo tipo de detalles sobre los dispositivos conectados a su red. En general, las compañías de telefonía móvil capturan y procesan una cantidad bastante elevada de información sobre sus usuarios. Esta información puede resultar muy valiosa si es tratada de forma correcta, puesto que puede serles útil para conocer en profundidad la actividad que se desarrolla en su red y adoptar decisiones en base a ello. Por tanto, el acceso a semejante volumen de información puede ofrecer numerosas ventajas que los propios operadores tratan de explotar al máximo.

Actualmente, los operadores manejan tres grandes bloques de información: datos sobre los dispositivos conectados, metadatos sobre paquetes que atraviesan la red e información sobre el contenido de los paquetes enviados y recibidos por los usuarios. Según el informe "*Retention Periods of Major Cellular Service Providers*" [5], publicado por la Unión Estadounidense por las Libertades Civiles a partir de la información proporcionada por el Departamento de Justicia de los EEUU, las compañías móviles se encuentran legalmente obligadas a proporcionar, por ejemplo, el registro de llamadas, los detalles y el contenido de los mensajes de texto intercambiados, el histórico de pagos e, incluso, los puntos de acceso utilizados para conectarse a la red. En este caso, esta información es accesible únicamente para fines jurídicos, pero hace visible la necesidad de mantener un control personalizado sobre la información específica de cada usuario.

Más allá de las cuestiones estrictamente legales, los operadores pueden utilizar la información a la que tienen acceso para sus propios fines, ya que tienen la posibilidad de detectar la llegada de cualquier dispositivo nuevo a la red e identificar, por ejemplo, su Sistema Operativo, su dirección IP, su consumo de ancho de banda e, incluso, las aplicaciones que se encuentra ejecutando. A efectos de los proveedores de servicios de Internet, la información individualizada de cada uno de sus usuarios no resulta especialmente relevante para ninguna razón particular, a excepción de determinar su consumo mensual para aplicarles las tarifas correspondientes. Sin embargo, el análisis del conjunto global de usuarios de la red puede permitirles comprender la utilización que se hace de ella en función de múltiples factores (horario, localización, grandes eventos…) y hacer un seguimiento para detectar tendencias y hábitos de consumo. De esta forma, se pueden hacer, por ejemplo, estimaciones sobre el tipo de contenidos que requerirán los usuarios en un periodo de tiempo concreto y reservar la cantidad de ancho de banda adecuada.

Toda esta información queda encapsulada cuando los usuarios conectan sus dispositivos móviles a redes Wifi. En ese caso, los operadores pierden el control sobre la actividad de sus usuarios y no son capaces de determinar, por ejemplo, qué dispositivos se están utilizando para navegar por internet, lo cual puede ser de interés para adaptar los contenidos que se muestran por pantalla durante la navegación. Para solucionar esta situación, en este Trabajo se ha estudiado el comportamiento de las redes de telefonía, tratando de identificar algún mecanismo que permita conocer con detalle qué elementos se encuentran conectados a una red, con el fin de implementar un mecanismo que automatice el reconocimiento de dispositivos en la misma.

En las redes inalámbricas actuales, existen dos opciones que permiten conocer qué elementos se encuentran en ellas:

- Detectar los dispositivos cercanos al Punto de Acceso (AP) → *Beacon Frames*.
- Analizar el tráfico HTTP generado → *User-Agent*.

Los *Beacon Frames* son un conjunto de tramas que se utilizan para la gestión de WLANs basadas en el estándar IEEE 802.11 [6]. Estas tramas se transmiten de forma periódica entre los Puntos de Acceso (AP) y los dispositivos localizados dentro de su rango y sirven para anunciar su presencia al extremo opuesto y enviar información sobre la configuración de la red. Por ejemplo, los AP utilizan estas herramientas para dar a conocer a los dispositivos cercanos su capacidad y la intensidad de señal óptima, permitiendo que las conexiones se realicen de la forma más adecuada. Además, en el caso de que finalmente se establezca una conexión entre extremos, ésta se sincroniza gracias a la información sobre la hora que existe dentro de este tipo de tramas.

Esta solución permite identificar dispositivos cuya posición es cercana al AP, pero no necesariamente conectados a él. Por tanto, esta opción no es válida para la propuesta llevada a cabo dentro de este Trabajo, ya que los dispositivos de interés son aquellos que se encuentran conectados a la red a través del router correspondiente. No obstante, mediante esta alternativa, se puede hacer una primera aproximación al caso de estudio e, incluso, se podría plantear como medio de trabajo si, por ejemplo, el enfoque dado al estudio consistiese en identificar el modelo de todos los dispositivos que se encuentran ubicados en una zona concreta.

Otra opción puede ser analizar el contenido de los paquetes generados por los dispositivos. En concreto, la información de interés para el ámbito de este Proyecto se encuentra en el User-Agent, localizado dentro de las cabeceras de los mensajes HTTP. La mayoría de dispositivos se encuentran constantemente conectados a servicios online en busca de actualizaciones, por lo que generan un tráfico que puede ser una fuente de información muy importante a la hora de buscar terminales móviles en la red.

La definición de este campo se encuentra en la sección 14.43 del RFC2616, titulado "Hypertext Transfer Protocol – HTTP/1.1" [7]. Según dicho documento, el User-Agent contiene información sobre el agente de usuario que realizó la petición HTTP contenida en el mensaje. Originalmente, su utilización estaba prevista para fines estadísticos, para la detección de violaciones del protocolo HTTP y para el reconocimiento automatizado de agentes de usuario, adaptando las respuestas enviadas en función de las limitaciones particulares de cada dispositivo. El campo puede contener múltiples *tokens* de producto y comentarios que identifiquen el agente y cualquier subproducto que forme una parte significativa de él.

Esta definición muestra que, inicialmente, el campo del User-Agent fue concebido para tres propósitos. El primero de ellos fue dar a las páginas web la posibilidad de conocer qué tipo de usuarios están navegando en ellas y utilizar dicha información para mostrarles los contenidos más adecuados en cada situación. También se tuvo en cuenta su posible ayuda frente a errores, ya que, en caso de fallos localizados, puede resultar útil para detectar si la limitación se debe a algún tipo de restricción particular con un agente dado. Por último, se planteó la posibilidad de emplearlo para adaptar las respuestas de los servidores en función de los agentes que las originan. Por ejemplo, si el navegador web que se emplea en una petición no tiene instaladas las extensiones correspondientes a *Adobe flash*, sería posible adaptar la web reemplazando el contenido flash por código *html* estático [8]. La tercera

opción es la más empleada en la actualidad; no obstante, el campo del User-Agent se puede emplear para muchas otras cuestiones que no se encontraban entre las identificadas originalmente.

Los datos incluidos dentro de los User-Agent incluyen información sobre los agentes que las generan, como, por ejemplo, el tipo de aplicación, el Sistema Operativo, el fabricante o la versión del software. Esta información es utilizada por muchos de los proveedores de servicios online, que pueden modificar el contenido que envían en función del tipo de dispositivos que están accediendo a ellos. Por ejemplo, algunos sitios web emplean esta información para mostrar contenidos específicos en dispositivos móviles.

Existen otras alternativas que permiten realizar una aproximación al estado de la red. Una de ellas consiste en analizar las direcciones MAC de cada dispositivo, puesto que dichas direcciones contienen cierta información que puede resultar de interés. Todos los dispositivos por cable e inalámbricos tienen configurada una o varias direcciones MAC, que son fijadas antes de su puesta en el mercado y se mantienen durante todo su ciclo de vida. Dentro de ellas, los tres primeros octetos identifican al fabricante de dicho dispositivo (Figura 2-1). Esta sección de la MAC se conoce como el *Organizationally Unique Identifier*, OUI, y es empleado por muchas aplicaciones para dar a los usuarios una información más completa en el análisis de los elementos de una red. Existen numerosos servicios online que cuentan con una base de datos completa sobre OUIs y permiten conocer cuál es el fabricante de un dispositivo a partir de su MAC. Uno de los más empleados es el que ofrece el analizador Wireshark en su página web [9], aunque existen muchas páginas similares.

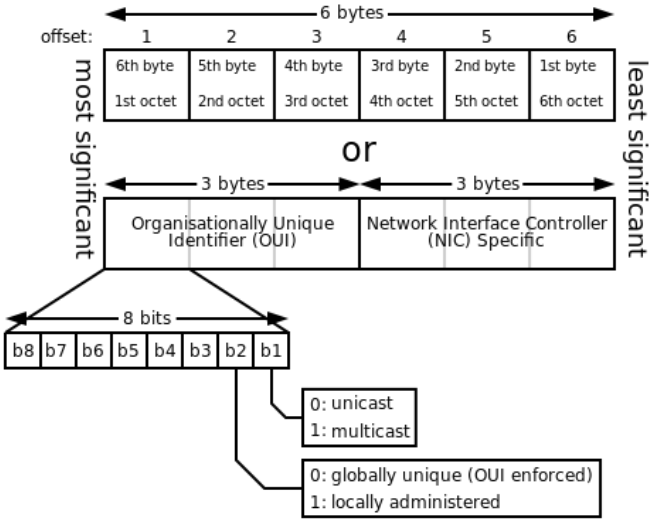

Figura 2-1. OUI de la dirección MAC.

(Fuente: *https://stackoverflow.com/questions/15989448/wireless-mac-address-patterns*)

2.2 Metodología empleada

2.2.1 DPI

Una de las técnicas a tener en cuenta para la realización de este Trabajo es el denominado *Deep Packet Inspection* (DPI) [10]. Se trata de un método avanzado de análisis de paquetes que se realiza sobre la capa de Aplicación del modelo de referencia TCP/IP. Su principal característica es que el análisis se realiza sobre toda la carga útil de los paquetes, examinando su contenido en busca de patrones que identifiquen el tráfico que está transportando. Esto incluye la mera detección de tipos de tráfico o aplicaciones, pero también violaciones de protocolo, virus, spam o cualquier otra restricción impuesta por los administradores de red [11].

La utilización de sistemas basados en *Deep Packet Inspection* sobre el tráfico de una red hace posible encontrar, identificar, clasificar, redireccionar o bloquear paquetes que contienen un tipo de datos concreto o algún patrón de código en su carga útil. Esta es una de sus principales ventajas, puesto que las técnicas convencionales de filtrado de paquetes, que sólo examinan sus cabeceras, no pueden detectar este tipo de información.

El análisis de los paquetes se efectúa en el momento en que éstos atraviesan el punto de inspección. En dicho punto, se lleva a cabo un control sobre ellos, buscando cadenas de caracteres o secuencias de bytes predefinidas que permitan implementar funciones relacionadas con la seguridad o con la minería de datos. Actualmente, numerosas empresas y proveedores de servicios emplean una elevada cantidad de recursos para desarrollar software que les permita aplicar las funcionalidades que ofrece el DPI [12].

Al trabajar directamente sobre la capa de Aplicación, es posible filtrar por programas o aplicaciones (por ejemplo, *Youtube* o *Skype*), dando la posibilidad a los administradores de red de conocer con mayor exactitud la actividad que se está desarrollando en ella. Su efectividad es alta frente a ataques de desbordamiento de buffer y de denegación de servicio (DoS). La técnica DPI también se puede utilizar para controlar y limitar las conexiones P2P (peer-to-peer), mejorando el rendimiento general de la red con las limitaciones que la progresiva encriptación entre clientes vaya imponiendo [13]. Por todo ello, las implicaciones de seguridad son bastante notables, puesto que es posible identificar la dirección IP del autor o receptor de contenido que incluya paquetes específicos y, potencialmente, correlacionarlo con la información de los operadores.

Pese a sus múltiples ventajas, es posible señalar algunas limitaciones relevantes de esta técnica. Por ejemplo, puede generar nuevas vulnerabilidades si el software sobre el que se lanza es atacado. Además, añade una carga considerable a los procesadores, reduciendo la velocidad de las máquinas que lo están implementando o, incluso, llegando a saturar potentes servidores a altas tasas. No obstante, ofrece un gran número de ventajas y su uso es cada vez más frecuente en todo tipo de redes.

2.2.2 HTTP

HTTP (HypertText Transfer Protocol) es el protocolo de comunicación empleado para las transferencias de información entre los elementos software que componen la arquitectura

web. Este protocolo se emplea para el intercambio de documentos HTML, archivos CSS, Javascript, imágenes y otros recursos similares. Permite la comunicación entre servidores, proxies y clientes, sin tener en cuenta cuáles son el cliente o servidor que realizan las peticiones.

El protocolo HTTP sigue un esquema petición-respuesta en el que el navegador web, que actúa como cliente, envía un mensaje de petición a un servidor web, que genera una respuesta ante dicha petición (Figura 2-2). Generalmente, entre ambos extremos hay puntos intermedios o *proxies*, que realizan distintas funciones, como, por ejemplo, actuar como *gateways* o como cachés. Existen también otros intermediarios, como routers o módems, pero de cara a servidores y navegadores web estos puntos son transparentes, puesto que HTTP se apoya en protocolos de las capas de red y transporte.

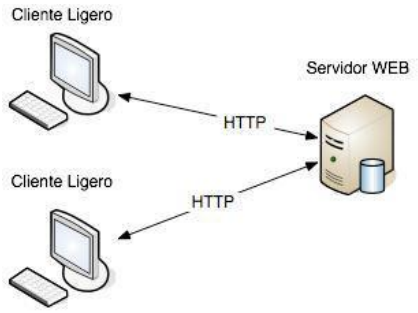

Figura 2-2. Interacción entre elementos software en el protocolo HTTP.
(Fuente: *http://roble.pntic.mec.es/jprp0006/tecnologia*)

En HTTP, las peticiones se componen de un conjunto de mensajes que se transmiten en texto plano. Las transmisiones se realizan sobre el protocolo TCP, o bien a través del protocolo encriptado TLS, aunque idealmente podría utilizarse sobre cualquier otro protocolo fiable. Diseñado a principios de los años 90, HTTP es un protocolo ampliable que ha ido evolucionando con el tiempo.

La definición de HTTP se encuentra en una serie de RFC, publicados por el IETF, entre los que se encuentra el RFC 2616 [7], en el que se define la versión HTTP/1.1 y se especifica la sintaxis y la semántica que deben utilizar los elementos software de la arquitectura web para comunicarse. Según este documento, es un protocolo de nivel de aplicación para sistemas de información distribuidos y colaborativos. Se trata de un protocolo genérico y sin estados que puede ser utilizado para muchas tareas más allá de su uso para el hipertexto, como el caso de servidores de nombres y sistemas distribuidos de gestión de objetos mediante la extensión de sus métodos de solicitud, códigos de error y cabeceras.

Cuando un extremo realiza una petición, genera un mensaje en el que se incluye:

- Una línea inicial, que contiene el método de solicitud, la URL del recurso solicitado y la versión del protocolo.
- Una lista de campos, entre los que se incluyen modificadores de la petición, información sobre el cliente, etc.
- Opcionalmente, un cuerpo con el contenido.

Por su parte, el servidor que recibe la petición se encarga de contestar al mensaje con una respuesta cuyo formato es similar al de la petición, aunque ajustándose a los parámetros solicitados (Figura 2-3).

Ejemplo de petición

```
GET /saludo.html HTTP/1.1
Host www.uam.es
```

Ejemplo de respuesta

```
HTTP/1.1 200 OK
Date: Sun, 01 May 2017 12:00:01 GMT
Content-Type: text/html
Content-Length: 45

<html>
        <body>Hola Mundo!</body>
</html>
```

Figura 2-3. Ejemplo de petición-respuesta mediante el protocolo HTTP.

HTTP es un protocolo sin estado, es decir, no almacena ningún dato entre peticiones dentro de la misma sesión. Para situaciones en las que pueda interesar seguir un flujo de navegación ordenado, existe la posibilidad de utilizar cookies, que permiten guardar datos en el sistema cliente de la sesión de comunicación. Como HTTP tiene entre sus propiedades la de la capacidad de ampliación, es posible hacer uso de cookies para crear un contexto común para cada sesión.

Las transmisiones de mensajes HTTP se apoyan en el uso de cabeceras o *headers*, que son metadatos que se envían en las peticiones y respuestas y que permiten intercambiar información esencial sobre la transacción en curso (Figura 2-4). La mayoría de las cabeceras HTTP son opcionales, por lo que es posible recibir respuesta del servidor incluso aunque se hayan omitido muchas de ellas. En el Anexo A.A se incluye un listado completo de todas las cabeceras posibles dentro del protocolo HTTP.

```
▼ Request Headers      view source
  Accept: image/webp,image/apng,image/*,*/*;q=0.8
  Accept-Encoding: gzip, deflate
  Accept-Language: es-ES,es;q=0.8
  Connection: keep-alive
  Cookie: __utmt=1; __utma=170659492.1522701672.1504042494.1504042494.1504042494.1;
   __utmb=170659492.4.9.1504042570758; __utmc=170659492; __utmz=170659492.150404249
  4.1.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none)
  Host: www.uam.es
  If-Modified-Since: Tue, 29 Aug 2017 05:00:04 CEST
  Referer: http://www.uam.es/UAM/Home.htm?language=es
  User-Agent: Mozilla/5.0 (Linux; Android 4.3; Nexus 10 Build/JSS15Q) AppleWebKit/53
  7.36 (KHTML, like Gecko) Chrome/60.0.3112.101 Safari/537.36
```

Figura 2-4. Ejemplo de cabeceras de una petición HTTP.

2.2.3 User-Agents

Uno de los campos contenidos entre las cabeceras de los mensajes HTTP es el correspondiente al User-Agent (UA). Este campo tiene por objeto identificar los dispositivos que solicitan contenido online. El UA indica al servidor qué tipo de dispositivo es el que ha realizado la petición (entre muchas otras cosas) y éste determina el contenido que enviará en la respuesta en función de dicha información. Por tanto, se requiere de una solución que permita detectar dispositivos y traducir los User Agent en información comprensible por el software que lo utiliza. Cuando las características de detección de APIs no se encuentran disponibles, los User-Agent sirven para personalizar el comportamiento o el contenido según la versión especificada del navegador web [14].

El formato de la cadena de texto incluida en el User-Agent se especifica en la sección 5.5.3 del RFC7231 [15]. Por diversas razones históricas, el contenido del User Agent incluye a menudo información que no es cierta sobre el dispositivo de origen. Además, puede contener palabras clave cuya finalidad es forzar un comportamiento concreto en el servidor web u obtener algún tipo de contenido deseado.

Según el estándar, el User-Agent se compone de uno o más *tokens* de producto que "identifican el software del agente de usuario y sus subproductos significantes". Los fabricantes de software utilizan estos *tokens* para enviar datos sobre el navegador, así como para obtener información específica sobre la ROM del dispositivo, el identificador del modelo, su Sistema Operativo y versión, etc.

Los *tokens* incluyen el nombre de producto y, opcionalmente, su versión. Habitualmente, se encuentran ordenados según su relevancia, aunque esta cuestión depende del navegador. También se especifica dentro del estándar que los UA deberían limitar el número de identificadores para aplicar únicamente aquéllos necesarios para identificar al producto en cuestión. Sin embargo, por lo general, no se siguen las reglas establecidas de manera estricta, especialmente la limitación en el número de identificadores. Además, la construcción de User Agents incorrectos no tiene ningún tipo de consecuencia más allá de que los dispositivos puedan no ser detectados correctamente. Por estas razones, el tratamiento del campo User-Agent puede resultar particularmente difícil.

A continuación, se muestran varios ejemplos con el contenido del campo User-Agent:

- Mozilla/5.0 (Linux; Android 6.0.1; **Aquaris X5** Build/MMB29M) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Mobile Safari/537.36
- Mozilla/5.0 (Linux; Android 6.0.1; **XT1562** Build/MPDS24.107-52-11) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.125 Mobile Safari/537.36
- Mozilla/5.0 (Linux; Android 4.3; **Nexus 10** Build/JSS15Q) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.101 Mobile Safari/537.36
- Mozilla/5.0 (iPhone; U; CPU **iPhone** OS **4_2_1** like Mac OS X; en-us) AppleWebKit/533.17.9 (KHTML, like Gecko) Version/5.0.2 Mobile/8C148 Safari/6533.18.5

Como se puede apreciar, el contenido de los User-Agent varía considerablemente en función del tipo de dispositivo que realiza la petición. Por tanto, si se desea utilizar esta información para identificar los terminales de una red, se hace necesario un análisis muy completo sobre su contenido.

2.2.4 Tablas Hash

Una tabla hash es un tipo de estructura que implementa un *array* asociativo de datos. Generalmente, se utilizan para almacenar un número elevado de registros. Las operaciones de búsqueda e inserción se realizan de una forma muy eficiente en este tipo de tablas, estableciendo una relación a partir de un conjunto de pares clave-valor. Cada clave debe ser única por cada registro dentro de las tablas, puesto que es el elemento a partir del cual se realiza la búsqueda de un determinado valor.

La información se almacena en posiciones pseudo-aleatorias (Figura 2-5), por lo que el acceso ordenado a su contenido es considerablemente lento, especialmente si el número de registros es elevado. Este tipo de implementación presenta un tiempo promedio de búsqueda menor que otras estructuras de datos, aunque en este caso la información no se encuentra distribuida siguiendo un orden que se pueda determinar de manera directa.

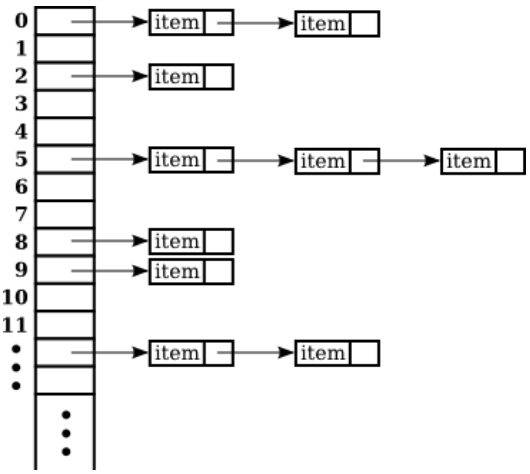

Figura 2-5. Implementación de una tabla hash.

(Fuente *http://math.hws.edu/eck/cs124/javanotes6/c10/s3.html*)

Cuando el número de registros a manipular es indeterminado, se necesita una estructura de datos dinámica. Las tablas hash ofrecen una alternativa muy interesante, puesto que permiten mantener un control sobre los datos y realizar operaciones sobre ellos de una manera eficiente.

Las implementaciones de tablas hash se basan en "funciones hash", que reciben la clave sobre la que se desea trabajar y devuelven un índice a partir de ella para determinar qué posición le corresponde en la tabla. Por lo general, se puede dar el caso de que varias claves devuelvan un mismo índice, lo cual se conoce como colisión. Ante esta situación, se

emplean mecanismos que resuelven el conflicto mediante la generación de posiciones distintas a la primera propuesta, utilizando, en caso necesario, búsquedas lineales.

Como los pasos seguidos son siempre los mismos, las posiciones resueltas para una misma clave son siempre idénticas, incluso en el caso de que se produzcan colisiones. Por tanto, este sistema garantiza que la búsqueda de elementos dentro de las tablas se realiza con fluidez y que su aprovechamiento es óptimo. De hecho, las tablas hash han sido de utilidad en múltiples sistemas de monitorización en entornos de red a alta velocidad [16].

En este Trabajo, las tablas hash se han utilizado para manejar la información relacionada con los dispositivos reconocidos y las firmas de terminales conocidos, facilitando la gestión de estos datos por parte de la aplicación. En el caso de la tabla encargada de almacenar los terminales identificados, el valor utilizado como clave es su dirección MAC, que lo identifica unívocamente. Por su parte, la tabla con las firmas conocidas utiliza la propia firma como clave, puesto que dos dispositivos distintos no van a tener el mismo contenido dentro de su User-Agent.

2.3 Trabajos relacionados

En la actualidad, existen en el mercado algunos trabajos que inspeccionan el tráfico generado por los dispositivos pertenecientes a una red para tratar de dar a los usuarios información útil sobre los elementos que la componen. Algunos tienen objetivos similares a los propuestos en este Trabajo, por lo que su estudio ha sido útil para conocer el estado actual del mercado y también para identificar algunas de las ideas propuestas en ellos y tenerlas como referencia para la implementación.

Por ejemplo, la aplicación ***AppPrint*** [17], desarrollada por un grupo de investigación de la Universidad de Texas y de la compañía *Narus Inc*., tiene por objetivo desarrollar un sistema que establezca firmas digitales a través de observaciones de tráfico completas e indique qué aplicaciones se encuentran utilizando los usuarios. La propuesta llevada a cabo en este caso se basa en el análisis de los User Agents de los paquetes generados por los usuarios para determinar las *apps* que cada usuario está utilizando. La principal diferencia respecto a este Proyecto es que, a pesar de emplear técnicas muy similares a las utilizadas en él, su propósito no está relacionado con la identificación de dispositivos móviles, sino de aplicaciones en uso.

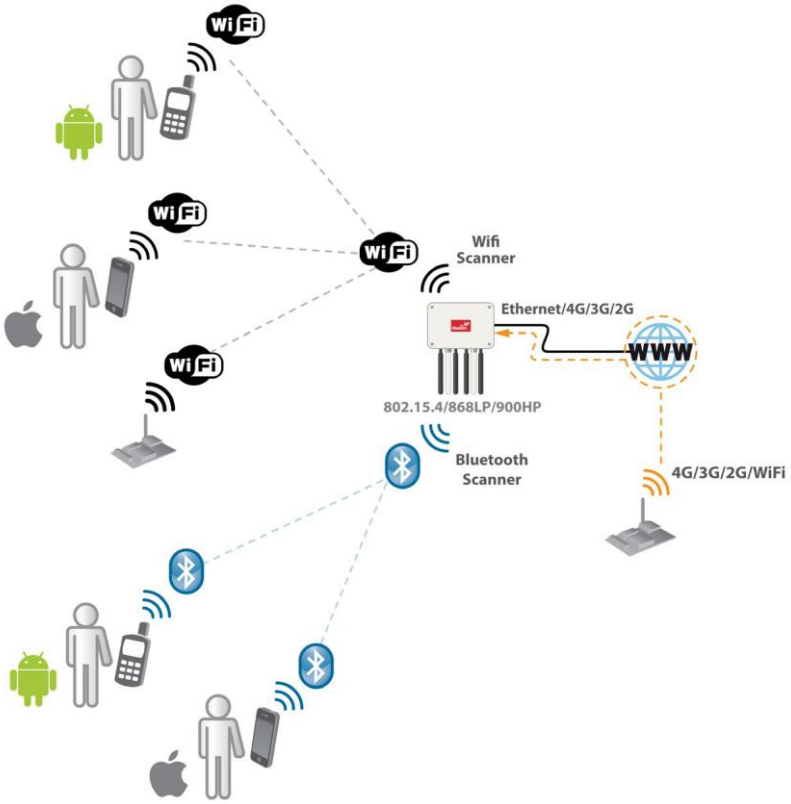

Figura 2-6. Meshlium Scanner 4G AP.

(Fuente: *http://www.libelium.com/products/meshlium/smartphone-detection*)

Existen otros programas, como ***Meshlium*** [18], desarrollado por la empresa *Libelium Comunicaciones Distribuidas S.L.*, que permite detectar el número de *smartphones* y, en general, cualquier dispositivo que cercano a través de sus interfaces Wifi y Bluetooth (Figura 2-6). La idea detrás de este sistema es poder determinar la cantidad de personas y de vehículos que están presentes en un punto en un instante específico, permitiendo estudiar la evolución del tráfico y la concurrencia a lo largo del tiempo e, incluso, el grado de congestión de una carretera a partir de la velocidad media de los vehículos que transitan en ella (Figura 2-7). Para ello, se hace uso de los *Beacon Frames* intercambiados entre los dispositivos y los AP para anunciar de manera recíproca su presencia. El programa muestra información de interés sobre los dispositivos detectados, como su dirección MAC, su intensidad de señal o su fabricante.

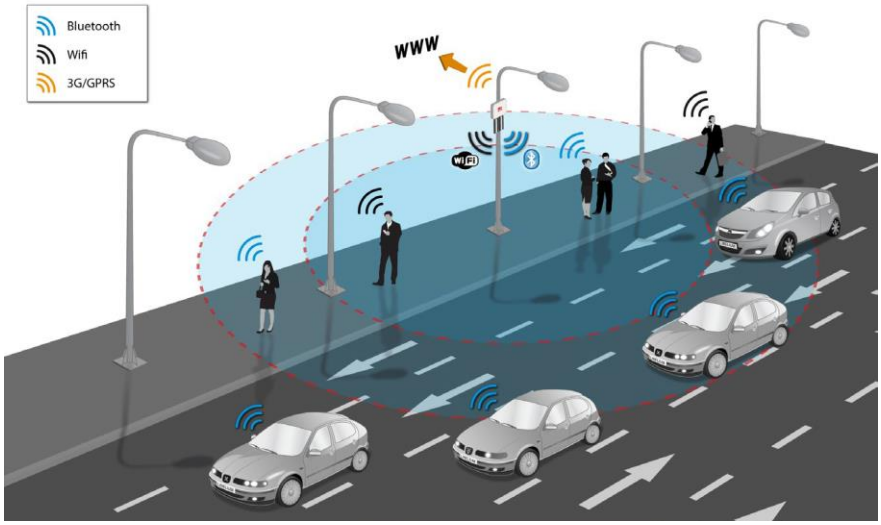

Figura 2-7. Meshlium: Vehicle Traffic Detection.

(Fuente: *http://www.libelium.com/products/meshlium/smartphone-detection*)

Por su parte, el software ***device-detection*** [19], creado por la empresa *51Degrees*, ofrece una solución de código abierto basada en una búsqueda de dispositivos optimizada a partir de su User-Agent, analizando cuestiones como el fabricante, el tamaño de pantalla o el navegador utilizado.

La propuesta desarrollada en otras herramientas software similares, como, por ejemplo, ***Detect Mobile Browsers*** [20], se basa en la inspección de los UA en busca de cadenas que indiquen si el Sistema Operativo pertenece a dispositivo móvil, pero no sé analiza de qué dispositivo se trata.

En general, programas como *Meshlium* o *device-detection* ofrecen a los administradores de las redes la posibilidad de conocer cuántos dispositivos se encuentran conectados a ella y cuál es su fabricante a partir del OUI de su MAC. Sin embargo, ninguna de las aplicaciones existentes ha sido desplegada en routers domésticos, tal y como se ha realizado en este Trabajo. Tampoco están orientadas a otorgar información útil para su tratamiento en tiempo real por parte de los operadores móviles ni permiten determinar qué modelos de dichos fabricantes se están empleando, siendo éste uno de los puntos clave de este Proyecto.

3 Descripción general de la aplicación desarrollada

La aplicación implementada pretende ser una herramienta fiable que aporte información de interés y actualizada a los gestores de red sobre qué dispositivos móviles se encuentran conectados a una red concreta. Para ello, se captura el tráfico intercambiado en la red a analizar y se vuelca la información generada en diferentes ficheros de salida, que pueden ser examinados durante la propia ejecución o después de la misma.

El *software* desarrollado es capaz de identificar los modelos de los dispositivos móviles que se están usando en la red de una forma transparente a los usuarios, es decir, sin tener que solicitar a los mismos su colaboración activa mientras la aplicación se encuentra en ejecución. Siguiendo una serie de pautas, se analizan los paquetes que se generan en los dispositivos de los usuarios mientras éstos se encuentran navegando, tratando de identificar patrones conocidos en ellos. Esta información es procesada y trasladada a los administradores de la red. Además, aparte del modelo identificado (o la propuesta de modelo, según el caso), se aporta información relevante sobre su actividad, como su dirección IP y MAC, su hora de llegada a la red, la cantidad de bytes de información generada, etc.

Está diseñada para ser desplegada en un router doméstico, permitiendo así conocer en tiempo real el uso que se está haciendo de la red por parte de los usuarios. El administrador de la red puede controlar en todo momento de qué forma está siendo utilizada y tomar decisiones en base a ello. La aplicación almacena el estado de cada elemento de la red y mantiene un histórico de los dispositivos que han estado conectados a ella, almacenando también datos sobre aquellos que ya la han abandonado.

Por defecto, la aplicación vuelca la información procesada en tres ficheros cada vez que se ejecuta:

- TRAFICO.log
  Se trata del fichero que contiene la lista completa con los paquetes capturados durante la ejecución de la aplicación, con información básica sobre cada uno de ellos. Además, en el caso de que se detecte que un paquete pertenece a un dispositivo móvil, se indica de qué modelo se trata.
  Este fichero permite a los usuarios de la aplicación tener un control sobre qué tráfico se ha generado en la red e, incluso, añadir o mejorar los patrones seguidos a la hora de determinar de qué dispositivo se trata cuando se detecta un nuevo elemento en la red (Figura 3-1).

```
248046 Puerto origen: 53390
248047 Puerto destino: 443
248048 Longitud: 429
248049 Hora: Wed Jul 19 23:26:48 2017
248050 Timestamp: 1500499608
248051
248052
248053 Paquete número: 20291
248054 IP origen: 192.168.187.152
248055 IP destino: 216.58.211.206
248056 MAC origen: 0:c:29:97:fa:dd
248057 MAC destino: 0:50:56:ef:12:a4
248058 Puerto origen: 53390
248059 Puerto destino: 443
248060 Longitud: 100
248061 Hora: Wed Jul 19 23:26:48 2017
248062 Timestamp: 1500499608
248063
248064
248065 Paquete número: 20292
248066 IP origen: 192.168.187.152
248067 IP destino: 212.113.89.75
248068 MAC origen: 0:c:29:97:fa:dd
248069 MAC destino: 0:50:56:ef:12:a4
248070 Puerto origen: 33648
248071 Puerto destino: 80
248072 Longitud: 531
248073 Hora: Wed Jul 19 23:26:48 2017
248074 Timestamp: 1500499608
248075 User-Agent: Mozilla/5.0 (Linux; Android 4.3; Nexus 7
       Build/JSS15Q) AppleWebKit/537.36 (KHTML, like Gecko)
       Chrome/42.0.2307.2 Safari/537.36
248076
248077 ***MODELO DETECTADO: Nexus 7***
248078
248079
248080
248081 Paquete número: 20293
248082 IP origen: 216.58.211.206
```

Figura 3-1. Captura de pantalla del fichero TRAFICO.log.

- DISPOSITIVOS_RECONOCIDOS.csv.
  En él se muestra la lista actualizada de dispositivos conectados a la red en el momento actual. Su contenido se actualiza cada 5 minutos (parámetro configurable) y permite identificar qué dispositivos utilizan los usuarios de la red y conocer datos relevantes sobre su navegación.
  Dentro de este archivo, aparecen, en primer lugar, los dispositivos "activos" dentro de la red, es decir, aquellos que se encuentran conectados en el instante en que éste ha sido generado, y, a continuación, aquellos que estuvieron previamente en ella, pero ya se han desconectado. Se dispone, por tanto, de un registro completo de los elementos de la red, tanto presentes como pasados (Figura 3-2).

MODELO	ACTIVO	WEB A VISITAR	DIRECCION IP	MAC	TIMESTAMP 1er PAQUETE	HORA 1er PAQUETE	TIMESTAMP ULTIMO PAQUETE	HORA ULTIMO PAQUETE	TOTAL BYTES INTERCAMBIADOS	MODELO CONOCIDO
BQ Aquaris X5	SI	https://www.google.es	192.168.1.121	73:bf:19:ae:21:fb	1499101274	Mon Jul 3 19:01:14 2017	1499102180	Mon Jul 3 19:16:20 2017	30117	SI
LG Optimus L70	SI	http://www.elmundo.es/	192.168.1.116	ab:12:c7:f1:23:ed	1499101157	Mon Jul 3 18:59:17 2017	1499102178	Mon Jul 3 19:16:18 2017	27323	SI
iPhone (versión no reconocida)	NO	htp://www.rae.es	192.168.1.117	d2:31:da:b5:8:83	1499100982	Mon Jul 3 18:56:22 2017	1499101705	Mon Jul 3 19:08:25 2017	4538	SI
HUAWEI H891L	NO	http://www.renfe.com	192.168.1.220	61:c:78:10:91:3f	1499100778	Mon Jul 3 19:52:58 2017	1499101265	Mon Jul 3 19:01:05 2017	5786343	NO
Nokia N9	NO	http://www.uam.es/UAM/Home.htm	192.168.1.114	a4:8d:ff:d2:40:51	1499100393	Mon Jul 3 19:46:33 2017	1499100464	Mon Jul 3 18:47:44 2017	1567680	SI

Figura 3-2. Captura de pantalla del fichero DISPOSITIVOS_RECONOCIDOS.csv.

- HISTORICO_DISPOSITIVOS.log
  En este fichero se almacenan, cada 300 segundos, los elementos conectados a la red en el momento actual. Durante la ejecución de la aplicación, se va añadiendo al final del fichero información sobre los dispositivos detectados. De esta forma, es posible acceder al histórico de dispositivos conectados y estudiar la evolución de los mismos a lo largo del tiempo (Figura 3-3).

```
-- DISPOSITIVOS RECONOCIDOS --
Thu Jul 20 00:06:55 2017

1. MODELO: Galaxy Note II
Web a visitar: http://m.20minutos.es/
MAC: 0:c:29:97:fa:dd
Dirección IP: 192.168.187.152
Timestamp 1er paquete: 1500499226
Hora 1er paquete: Wed Jul 19 23:20:26 2017
Timestamp último paquete: 1500501969
Hora último paquete: Thu Jul 20 00:06:09 2017
Total bytes intercambiados: 4301533
Modelo conocido

2. MODELO: iPhone (versión no reconocida)
Web a visitar: http://www.hpch-uam.es/
MAC: 0:50:56:ef:12:a4
Dirección IP: 150.244.214.237
Timestamp 1er paquete: 1500499364
Hora 1er paquete: Wed Jul 19 23:22:44 2017
Timestamp último paquete: 1500501850
Hora último paquete: Thu Jul 20 00:04:10 2017
Total bytes intercambiados: 18377
Modelo desconocido. Por favor, revise si es correcto y
actualice las tablas con las firmas conocidas.

3. MODELO: Nexus 7
Web a visitar: http://www.uam.es/UAM/Home.htm
MAC: 0:d:29:54:c3:f5
Dirección IP: 192.168.187.153
Timestamp 1er paquete: 1500492512
Hora 1er paquete: Wed Jul 19 23:25:12 2017
Timestamp último paquete: 1500501989
Hora último paquete: Thu Jul 20 00:06:29 2017
Total bytes intercambiados: 592980
Modelo conocido
```

Figura 3-3. Captura de pantalla del fichero HISTORICO_DISPOSITIVOS.log.

En todos estos ficheros se indica el modelo que se ha identificado para cada dispositivo detectado. También se incluyen las direcciones IP y MAC del dispositivo, la hora y el *timestamp* del primer paquete generado y del último paquete recibido hasta el momento, así como la web que se está visitando. Además, en los dos últimos ficheros se indica, por ejemplo, si el dispositivo se encuentra actualmente en la red, si se trata con certeza de un dispositivo móvil o si el modelo forma parte de la base de datos de firmas conocidas de la aplicación, incluyéndose también la longitud total de los paquetes generados durante la ejecución.

La captura del tráfico en el router se realiza sobre su interfaz inalámbrica, ya que es aquella se reciben directamente los paquetes de los dispositivos conectados a él. Los paquetes recibidos en esta interfaz no han sido aún procesados por el router, por lo que no contienen entre sus cabeceras las correspondientes a la capa VLAN, que hace que, llegado el caso, no se conserven, por ejemplo, las direcciones IP y MAC originales del dispositivo. Si, por ejemplo, se quisiese trabajar sobre una máquina virtual y actuar sobre los paquetes generados en el navegador, lo cual es útil para realizar pruebas en un entorno local, la interfaz que habría que escoger sería la cableada, que corresponde con la interfaz virtual de salida de la máquina virtual.

La identificación de los dispositivos se basa en tres grandes módulos funcionales, que dependen del Sistema Operativo del mismo: Android, iOS y otros. Las características concretas de cada uno de ellos hacen que la identificación deba hacerse aplicando lógicas distintas en cada caso. El contenido de los paquetes recibidos es analizado en busca de los patrones que indican cuál es el SO que los ha generado y, en función de los resultados obtenidos, se trabaja con el módulo adecuado. En el prototipo desarrollado, se ha optado por centrar los esfuerzos en dar soporte a los dos principales Sistemas Operativos del mercado (Android e iOS), puesto que, actualmente, entre ambos cuentan un 97,47 % de la cuota de mercado [21]; no obstante, se han tenido en cuenta todas las posibilidades existentes de cara a la identificación final, así como un proceso de aprendizaje de la herramienta frente a equipos inicialmente no reconocidos.

De esta forma, la aplicación cuenta con una base de datos de dispositivos conocidos que puede ser gestionada por el administrador de la red. Cuando se detecta la llegada de un nuevo elemento, el programa comprueba, en primer lugar, si la firma identificada coincide con alguno de los modelos conocidos de la base de datos. En caso afirmativo, se toma su valor en las tablas como modelo válido y se indica en los ficheros de salida que se trata de un modelo conocido. Por el contrario, cuando el nuevo terminal no se encuentra en las tablas, se propone cuál podría ser su modelo a partir del contenido del paquete y se solicita al usuario final que revise si es correcto para actualizar las tablas con las firmas conocidas. En este caso, la persona encargada de mantener la aplicación tendría la opción de añadirlo a las tablas de dispositivos conocidos para dar completitud a la aplicación y que, en futuras ejecuciones, dicho dispositivo fuese reconocido de manera directa. La versión actual de la aplicación cuenta con alrededor de 30 firmas conocidas, aunque esta información podrá ser completada a partir de las observaciones realizadas durante su uso.

Las estructuras de datos han sido implementadas mediante tablas hash, aprovechando el potencial que ofrece este tipo de estructuras para realizar operaciones de búsqueda e inserción de datos. La dirección MAC del dispositivo es el elemento que se utiliza como clave para evaluar si éste ya se encuentra en las tablas de dispositivos reconocidos, ya que dicha dirección es única y permite identificarlo de manera unívoca. Si ya había sido identificado, simplemente se actualiza la información correspondiente a la hora de llegada

del último paquete, la página web a visitar y el número total de bytes de los paquetes generados; si, por el contrario, se trata de un nuevo elemento dentro de la red, se crea una nueva entrada en las tablas y se inicializan todos los valores en función del contenido de dicho paquete.

Otra de las cuestiones que se ha tenido en cuenta es que, según el análisis que se hizo sobre el contenido de los paquetes HTTP generados por los dispositivos, existe una palabra clave dentro del campo User-Agent que puede ayudar a determinar si se trata o no de un terminal móvil: "*Mobile*". Conociendo esta información, se hace una búsqueda dentro del payload de los paquetes recibidos para, en el caso de que la firma identificada no coincida con la de ningún dispositivo conocido, indicar al usuario final que dicho dispositivo podría no ser un teléfono móvil, dando así más completitud a la información que manejará el administrador de la red.

El *software* desarrollado también permite analizar trazas de tráfico offline, otorgando cierto grado de versatilidad a la aplicación. Además de poder examinar el tráfico "en vivo", la versión actual permite estudiar la evolución de una red a partir de una traza ya generada, lo cual puede ser de gran utilidad, principalmente, en entornos de pruebas. El prototipo implementado está preparado para que, con un único cambio en el código, el cambio entre las dos modalidades (tráfico online u offline) sea directo.

4 Metodología general

4.1 Herramientas empleadas

Las herramientas empleadas pueden separarse en dos grandes grupos, según la parte a la que pertenecen: *hardware* o *software*.

Herramientas Hardware.

- **Ordenador**.
  Utilizado para la implementación de los ficheros de los que consta la aplicación y, en general, para todos los aspectos relacionados con el trabajo previo al desarrollo de la misma y con el posterior análisis de los datos originados con su ejecución. En este caso, la máquina empleada fue un ordenador portátil de uso personal.

- **Router doméstico**.
  Necesario para capturar el tráfico en un entorno real. El modelo utilizado fue el ***Cisco Linksys E3000***, disponible en la Escuela Politécnica Superior para su uso académico.

Herramientas Software.

- **Sistema Operativo Ubuntu (Linux).**
  A través de una Máquina Virtual creada e instalada mediante la herramienta VMware Workstation 12 Player, se empleó una versión de este Sistema Operativo, basado en GNU/Linux, como base para realizar toda la implementación del código de la aplicación.

  Dado que el desarrollo realizado no tenía demasiados requerimientos a nivel de procesamiento o almacenamiento, las capacidades reservadas fueron las que se consideraron razonables dentro de los límites establecidos por el software y las condiciones concretas del PC utilizado. Las características básicas escogidas para la configuración de la máquina virtual fueron las que se muestran en la Figura 4-1.

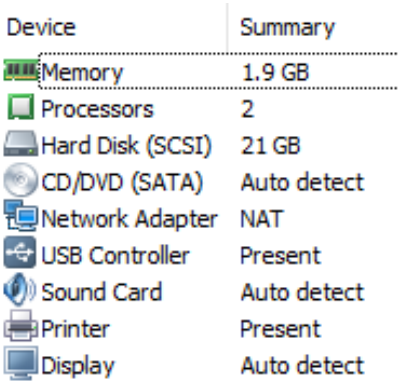

Figura 4-1. Características de la VM Ubuntu instalada.

- **Herramientas de edición y desarrollo**.
  El editor de textos empleado fue ***gedit***, instalado por defecto en la máquina virtual utilizada. Se trata de un editor de propósito general, compatible con múltiples Sistemas Operativos y que incorpora herramientas para la edición de código fuente en múltiples lenguajes de programación.

- **Entorno de pruebas software.**
  Para realizar las pruebas oportunas se utilizó ***Selenium*** [22], que proporciona un entorno de pruebas software para aplicaciones y programas basados en la web. Para este Trabajo, se empleó para simular un usuario abriendo un navegador y consultado diferentes páginas web en Internet.

- **Navegadores web para el entorno de pruebas.**
  En este Trabajo, se optó por utilizar dos navegadores web diferentes durante la fase de desarrollo: ***Chromium*** y ***Mozilla Firefox***. Ambos navegadores se encuentran ampliamente extendidos en el mercado y, como consecuencia de ello, son muy potentes y proporcionan un nivel de soporte a sus usuarios bastante notable.

  Además, en los dos casos, cuentan con una serie de herramientas para desarrolladores muy completas y que son de gran utilidad, puesto que, entre otras cuestiones, ofrecen la posibilidad de simular el funcionamiento de distintos dispositivos móviles reales y analizar la forma en que se visualiza el contenido en pantalla (Figura 4-2). Esta opción permitió disponer de un grupo interesante de User Agents y terminales como conjunto de validación y como entrada inicial a la aplicación.

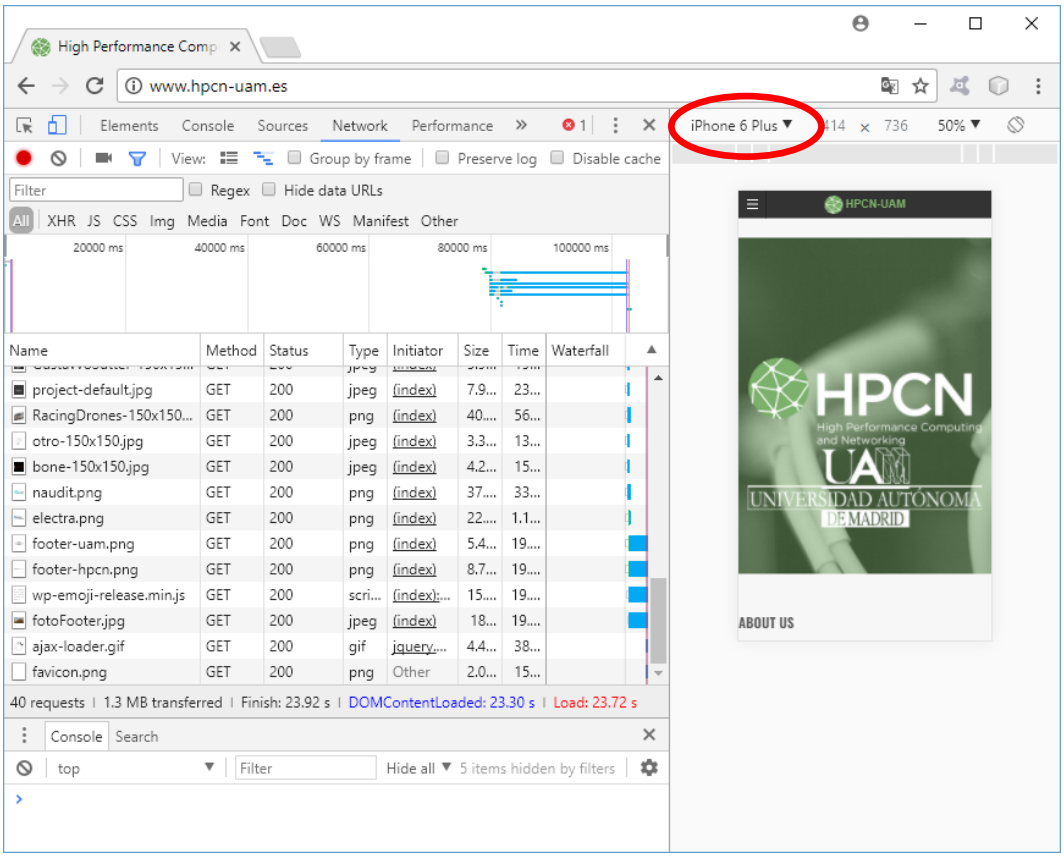

Figura 4-2. *Toogle Device Toolbar* en navegador *Chrome*.

- **Capturador de paquetes.**
  Para capturar el tráfico generado por los dispositivos que se encuentran conectados a la red, se hizo uso de las librerías de ***libpcap*** [23]. A través de estas librerías, la aplicación filtra y procesa los paquetes intercambiados dentro de la red, haciendo posible su posterior tratamiento.

- **Compilador cruzado.**
  El proceso de compilación de los ficheros de los que se compone la aplicación se realizó mediante ***Buildroot*** [24]. De esta forma, se consiguió que el programa pudiese ser ejecutado en arquitecturas diferentes de la utilizada para compilarlo. En este caso, la arquitectura de destino sería la del router doméstico, que es donde comercialmente se ejecutaría el desarrollo.

- **Herramientas de control de versiones.**
  Durante la fase de desarrollo, se mantuvo un control de versiones utilizando ***SVN***. Esta herramienta permite mantener un histórico sobre las modificaciones realizadas y poder comparar y revertir los cambios en el caso de que sea necesario.

- **Aplicación para capturar en local el tráfico generado en un terminal móvil.** Por último, para realizar capturas de tráfico sobre dispositivos móviles reales que permitiesen analizar los paquetes generados antes de comenzar con el desarrollo de la aplicación en sí, se utilizó la aplicación para dispositivos Android ***tPacketCapture*** [25], disponible de manera gratuita en *Play Store*. Esta aplicación captura el tráfico que genera el dispositivo de forma local y vuelca la información en un fichero con extensión .pcap, de forma que es posible analizarlo después de la ejecución y estudiar algunas de sus características.

4.2 Plan de trabajo

Los esfuerzos durante el desarrollo se centraron en cumplir los objetivos marcados a corto/medio plazo en cada una de las reuniones fijadas de forma periódica. Se podrían identificar las siguientes fases durante el desarrollo:

- Revisión del estado del arte.
  En primer lugar, se realizó un estudio en profundidad sobre el funcionamiento de las redes de telefonía y se investigó acerca del estado de trabajos relacionados con la idea desarrollada en este Proyecto, con el fin de conocer algunas de las propuestas llevadas a cabo en propuestas similares y examinar las técnicas empleadas para la identificación de modelos de dispositivos móviles.

- Análisis de características del contexto.
  Una vez estudiadas las distintas alternativas para clasificar a los diferentes dispositivos, se procedió a realizar una propuesta sobre la estrategia a seguir en la aplicación para alcanzar el objetivo final, determinar inequívocamente los modelos de los dispositivos conectados a una red. Para ello, fue necesario realizar una inspección manual del tráfico que generan los diferentes terminales móviles existentes en el mercado para tratar de inferir pautas que permitiesen identificar cada uno de ellos. Por tanto, se analizó a fondo el comportamiento del tráfico en función del Sistema Operativo y fabricante de cada dispositivo.

- Diseño e implementación.
  Durante este periodo, se desarrollaron las herramientas y módulos que permitieron simular la utilización de dispositivos móviles, capturar el tráfico generado, analizar la carga útil de los paquetes, almacenar información de interés en tablas hash y volcar los contenidos en ficheros de salida. El funcionamiento fue probado mediante navegadores web a través del entorno de pruebas que ofrece *Selenium*. Esta fase fue la de mayor peso dentro del plan de trabajo, puesto que en ella se realizaron las operaciones de mayor importancia de cara al resultado final.

- Integración de todas las herramientas en una aplicación.
  A continuación, dichas herramientas pasaron a formar parte de una única aplicación global, que incorporaba todas las funcionalidades requeridas y que es la que más tarde se instaló en el router sobre el que se realizaron las pruebas.

- Validación de uso.
  A partir de la puesta en marcha de la aplicación en el entorno de pruebas, se evaluó su funcionamiento comparando los datos generados con los que se podrían esperar del escenario sobre el que se realizaron las mismas, analizando así el comportamiento de una serie de dispositivos conocidos.

- Instalación del programa en un entorno real.
  Tras realizar las pruebas oportunas en el entorno de simulación, el siguiente paso fue trasladar la aplicación a un contexto real. Para ello, fue necesario configurar previamente el router sobre el que llevaron a cabo dichas pruebas. Se realizó un análisis detallado del comportamiento del programa, con el fin de detectar posibles fallos y solucionarlos de cara a la versión final del mismo, actualizada y corregida.

- Análisis de los resultados.
  Con los datos obtenidos, se verificó la eficacia de la aplicación y, en base a ello, se extrajeron las conclusiones correspondientes sobre los resultados. Además, se identificaron algunos mecanismos que podrían permitir mejorar su eficacia en versiones futuras de la aplicación.

4.3 Desarrollo

4.3.1 Documentación previa

La aplicación ha sido implementada en su totalidad en este Trabajo. El código del programa realizado ha sido realizado directamente durante el desarrollo del mismo, puesto que no se partía de ningún otro proyecto previo ni se continuaba con las propuestas definidas en estudios similares. Esto implicó la necesidad añadida de tener que llevar a cabo una aproximación previa al caso de estudio para comprender en profundidad el comportamiento de las redes de comunicaciones en función de la utilización que hacen de ellas sus usuarios.

Antes de la implementación del código de la aplicación, se llevaron a cabo diversas pruebas básicas con el fin de comprender a fondo el ámbito del Trabajo. Estas pruebas sirvieron, entre otras cuestiones, para identificar distintas alternativas a la hora de conocer el modelo exacto de un dispositivo.

Una de las posibilidades para averiguar a qué modelo pertenece un terminal es la instalación de una aplicación dedicada en él. El Sistema Operativo *Android* ofrece la opción de conocer el modelo y el fabricante de un dispositivo mediante dos propiedades que se pueden incluir en el código de las aplicaciones desarrolladas para esta plataforma [26]. En la Figura 4-3 se muestran las propiedades que se emplean para conocer dicha información en dispositivos Android.

```
String deviceName = android.os.Build.MODEL;
String deviceMan = android.os.Build.MANUFACTURER;
```

Figura 4-3. Propiedades Android para conocer el modelo y fabricante de un modelo.

A partir de estos datos, se implementó una aplicación muy simple de prueba (*PruebaDeviceModel*) que permitió determinar la fiabilidad de estas características. La aplicación desarrollada simplemente muestra por pantalla el modelo y el fabricante del dispositivo desde el que está siendo lanzada.

En la Figura 4-4 se pueden observar los resultados tras la ejecución de esta *app* de prueba en un terminal móvil. Dichos resultados coincidieron con los esperados en todas las pruebas realizadas, por lo que se puede afirmar que la fiabilidad de estas instrucciones es muy elevada.

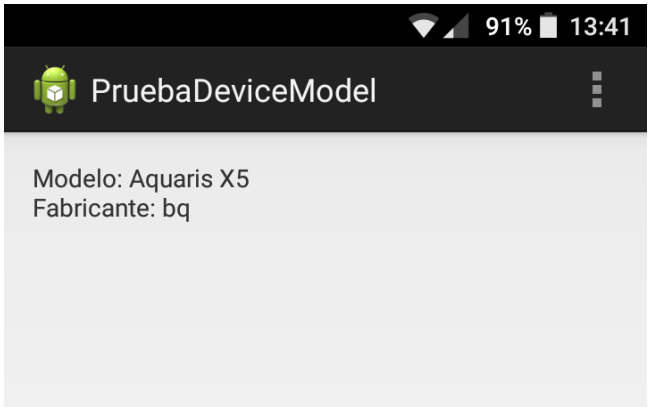

Figura 4-4. Captura de pantalla de la aplicación de prueba *PruebaDeviceModel*.

No obstante, esta posibilidad se descartó por el carácter intrusivo de su planteamiento, puesto que esta alternativa obligaría a todos los usuarios a tener instalada una aplicación concreta en su teléfono y contar con su colaboración. Sin embargo, sirvió para realizar una primera aproximación a las posibilidades existentes y para comprobar que, en caso necesario, se podrían extraer datos relevantes incluyendo simplemente en cualquiera de las aplicaciones del usuario las dos líneas indicadas en la Figura 4-3.

La siguiente alternativa que se planteó fue analizar el tráfico producido por los terminales en busca de patrones o pautas que ayudasen a determinar qué dispositivos habían generado los paquetes capturados. Esta solución no plantea la necesidad de que el usuario tenga constancia de que los operadores móviles están realizando un seguimiento de su actividad, puesto que la captura se realiza de forma transparente a ellos, tal y como ocurre al conectarse a las redes de telefonía.

Para tratar de encontrar las pautas que permitiesen reconocer un modelo respecto al resto, se realizó un análisis muy completo de los paquetes intercambiados por distintos dispositivos. Se tuvieron en cuenta múltiples características que podrían identificar de manera unívoca a cada tipo de dispositivo y se estudiaron en detalle todas ellas. Entre las cuestiones que se analizaron, se puede desatacar los tamaños de pantalla de los dispositivos, los diferentes protocolos utilizados, sus métricas, los campos que contienen, etc.

Inicialmente, el tráfico fue capturado de forma local utilizando los emuladores proporcionados por los navegadores Chrome y Firefox en su versión para desarrolladores. Sobre las capturas realizadas se pudieron observar, a partir de la inspección manual de los paquetes, los diferentes patrones que sigue el tráfico que generan los dispositivos y llegar a

la conclusión de que, de entre todas las características estudiadas, la que mejor se ajustaba al caso de estudio era el análisis del tráfico HTTP y, más concretamente, del campo *User-Agent* contenido entre las cabeceras de los paquetes de dicho protocolo. Esta averiguación resultó clave para la evolución del Trabajo, puesto que en ella se basa el prototipo desarrollado.

Por otra parte, para corroborar que el tráfico de los emuladores cumplía con el mismo formato que el de los dispositivos reales y, al mismo tiempo, para disponer también de una traza real sobre la que realizar pruebas, se utilizó una aplicación para capturar tráfico en terminales Android: *tPacketCapture*. Esta aplicación, disponible en el *Play Store* de forma gratuita, permite observar todos los paquetes que se generan en el móvil o *tablet* desde el que se está lanzando, haciendo posible trabajar sobre datos reales generados mientras los usuarios se encuentran navegando en la web. La Figura 4-5 muestra el aspecto de las distintas ventanas de las que se compone la aplicación.

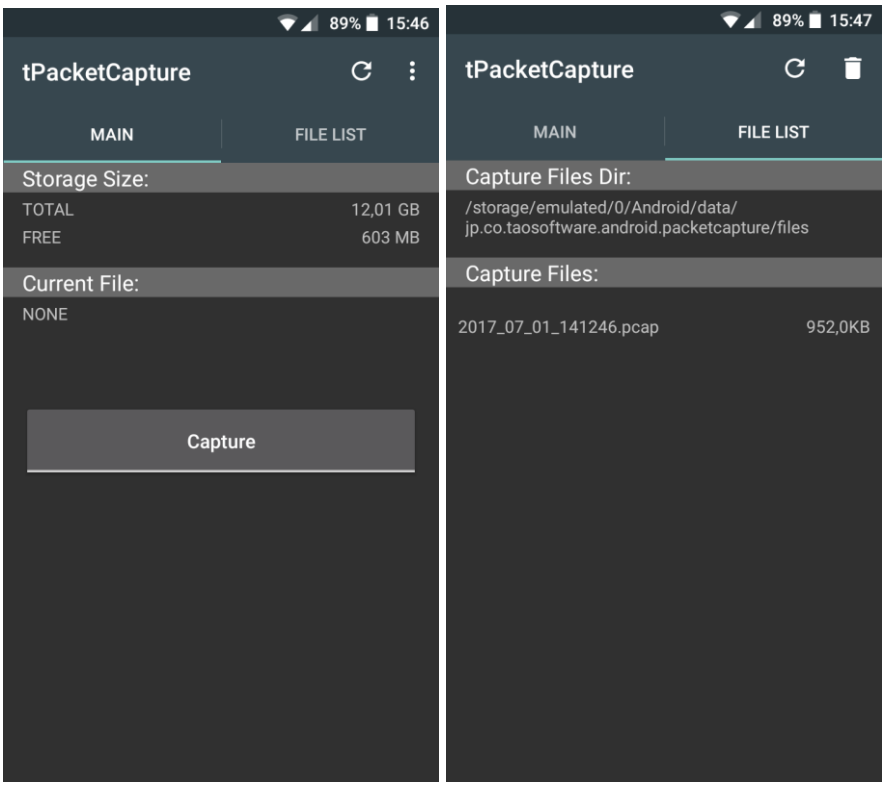

Figura 4-5. Ventanas de la aplicación *tPacketCapture*.

Tras el análisis de las capturas de *tPacketCapture*, se pudo determinar que los emuladores de navegadores como Firefox o Chrome se ajustan en gran medida al funcionamiento de dispositivos reales, incluso en términos de sus peticiones HTTP. Con toda la información recopilada, se comenzó con la fase de recopilación de User Agents de diferentes terminales, con el fin de contar con una base de datos sobre la que comparar las identificaciones realizadas.

4.3.2 Cuestiones específicas

A lo largo de todo el desarrollo llevado a cabo para la aplicación, se mantuvo un control de versiones utilizando las herramientas proporcionadas por *Apache Subversion* (SVN). Se trata de un software libre que hace accesibles las diferentes versiones de los archivos del programa mediante un conjunto de repositorios similar al de un sistema de ficheros. Ofrece un elevado grado de flexibilidad y permite mantener un control bastante directo sobre la evolución del código durante su implementación, lo que resulta de gran utilidad para desarrollos similares al de este Trabajo.

Este software efectúa un seguimiento sobre el historial de versiones de los ficheros de su repositorio y permite visualizar las modificaciones realizadas en cada momento, así como recuperar los cambios en caso de error. Además, está diseñado para trabajar de manera grupal, puesto que cuenta con repositorios accesibles de manera remota y está preparado para que los diferentes cambios realizados por las personas dentro del grupo de trabajo no generen conflictos; no obstante, estas funcionalidades no fueron utilizadas en este Trabajo por sus características concretas. Por todas las razones mencionadas, se optó por emplear SVN como elemento base para el control de versiones, puesto que es una herramienta muy potente y, a su vez, de fácil uso para el ámbito específico de la aplicación (Figura 4-6).

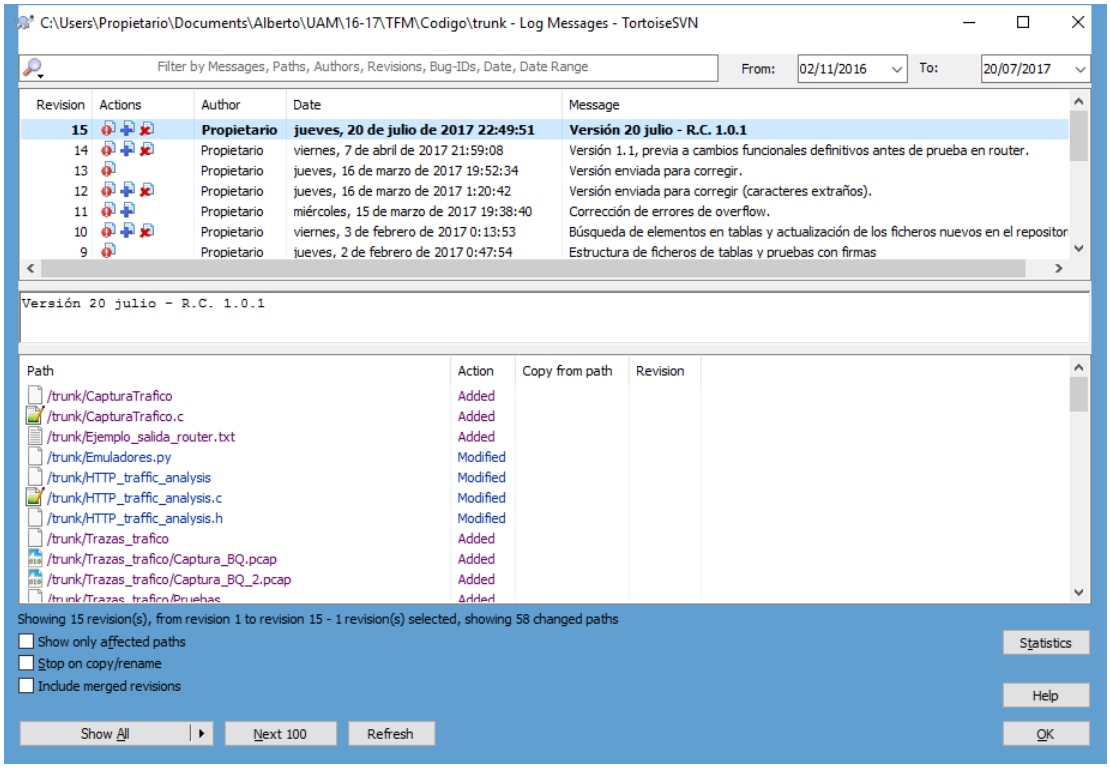

Figura 4-6. Histórico de versiones en SVN.

El proceso de captura de tráfico dentro de la aplicación se realiza mediante las librerías de *libpcap*. Se trata de un conjunto de librerías *open source* escritas en el lenguaje de programación C que ofrecen la posibilidad de disponer de una interfaz desde la que capturar paquetes de la capa de red. Su implementación está desarrollada para sistemas basados en Unix, aunque una de sus principales ventajas es que es portable entre un gran número de Sistemas Operativos [27]. Mediante esta herramienta, se realiza en la aplicación

desarrollada la lectura de ficheros con trazas offline, así como la captura, el filtrado y la lectura de los paquetes de bajo nivel intercambiados en la red.

Los paquetes capturados son analizados en busca de patrones conocidos, con el fin de poder determinar qué dispositivo ha generado cada uno de ellos. Para ello, durante la ejecución se estudia el contenido de su carga útil y se realiza un procesamiento del texto incluido entre sus cabeceras con el fin de extraer la información relevante que existe dentro del campo del User-Agent.

Los User-Agent, definidos en el RFC2616, se utilizan para dar información sobre los dispositivos que generan las peticiones. Como se comentó en el Capítulo 2.2.2, el formato de dicho campo no está estandarizado y los fabricantes no siguen siempre las mismas reglas. Antes de la implementación de la aplicación, se realizó un estudio previo en busca de patrones que permitiesen determinar qué características identifican a cada dispositivo. El formato de los UA viene marcado, principalmente, por su Sistema Operativo y fabricante, puesto que, en función de estos parámetros, las cabeceras HTTP siguen unas u otras reglas.

Para poder identificar un dispositivo, es necesario conocer cómo se componen los User-Agent. A continuación, se expone el análisis del ejemplo concreto de un dispositivo con Sistema Operativo *Android*.

```
Mozilla/5.0 (Linux; U; Android 4.3; en-us; SM-N900T Build/JSS15J)
AppleWebkit/543.30 (KHTML, like Gecko) Version 4.0 Mobile
Safari/534.30
```

El User-Agent se divide en las siguientes partes:

- ***Mozilla.*** UA basado en *Mozilla*. Esto únicamente es cierto para navegadores con motor de renderizado *Gecko*. Para el resto, significa que es compatible con *Mozilla*. En los navegadores actuales, no tiene significado real.
- ***5.0.*** Versión de *Mozilla* utilizada.
- ***Linux.*** Sistema Operativo sobre el que está montado el dispositivo.
- ***U.*** Valor correspondiente al nivel de seguridad.
    - N – No security.
    - U – Strongly secured.
    - I – Weak security.
- ***Android 4.3.*** Sistema Operativo y versión del dispositivo.
- ***en-us.*** Codificación detectada (por ejemplo, en menús y botones en la interfaz).
- ***SM-N900T.*** MODELO DEL DISPOSITIVO.
- ***Build/JSS15J.*** Número de compilación o *build number* del dispositivo Android.
- ***AppleWebkit/534.30.*** Versión de la plataforma para aplicaciones *Webkit*, que funciona como base para navegadores como *Safari, Opera…* Proporciona un conjunto de clases básicas para mostrar el contenido.
- ***KHTML.*** Motor de diseño HTML Open Source desarrollado por el Proyecto KDE.
- ***Version 4.0.*** Versión de *Safari*.
- ***Mobile.*** INDICADOR DE DISPOSITIVO MÓVIL.

- ***Safari/534.30.*** Número de compilación de *Safari*.

Según esta composición, la parte de interés para este Trabajo se encuentra en dos de sus apartados: el indicador "*Mobile*", que puede ayudar a determinar con certeza si el dispositivo en cuestión es o no un terminal móvil, y, principalmente, la cadena con el modelo del dispositivo (en este caso, SM-N900T).

No obstante, la cadena de texto que representa al dispositivo concreto no contiene literalmente su modelo exacto, sino un identificador. Dependiendo de su fabricante, esta cadena se asemeja en mayor o menor medida al modelo real del teléfono o *tablet* analizado. Algunos fabricantes optan por especificar directamente el modelo de dispositivo en dicho apartado, incluyéndolo tal cual en la cabecera; otros emplean su alias comercial o, simplemente, otra información que lo identifica, como ocurre en el ejemplo propuesto, en el que la cadena SM-N900T equivale al teléfono móvil *Galaxy Note 3*, fabricado por la compañía *Samsung*.

Ante esta circunstancia, se pudo establecer un diccionario de firmas conocidas que determine la relación entre el texto encontrado dentro de los User-Agent y los modelos a los que identifican. La aplicación desarrollada cuenta con una tabla en su base de datos con algunas firmas conocidas, de forma que, cuando se detecta un nuevo dispositivo, se comprueba si la cadena corresponde a uno de los terminales predefinidos, en cuyo caso se determina con exactitud el modelo concreto del que se trata. Sin embargo, si dicha firma no se encuentra en las tablas existentes, se muestra en los registros la firma completa y se indica que no se conoce su modelo exacto, dejando la posibilidad al administrador de la red de añadirlo a la base de datos de la aplicación para futuras ejecuciones.

Para realizar las pruebas oportunas durante la implementación de la aplicación, se desarrolló un pequeño programa, codificado mediante el lenguaje de programación Python, que simulaba una serie de visitas a diferentes páginas web desde un conjunto de terminales móviles predefinidos. Este archivo, que se encuentra entre los ficheros incluidos dentro de la entrega final de este Trabajo con el nombre de ***Emuladores.py***, tenía por objeto simular el comportamiento de distintos terminales que se encontraban navegando por Internet durante periodos de tiempo relativamente cortos. De esta forma, se consiguió generar tráfico que podía ser capturado en tiempo real, manteniendo además las características concretas de los paquetes generados en dispositivos reales.

Este fichero carga progresivamente la configuración de distintos dispositivos previamente establecidos y emplea el denominado "Toogle Device Toolbar", disponible entre las Herramientas para Desarrolladores que ofrece el navegador Chrome [28], para emular la utilización de terminales reales. En cada ejecución, se recorre la lista de dispositivos y se genera una visita a una página web aleatoria escogida dentro de un conjunto de webs fijadas en el fichero. Para este Trabajo, se incluyeron tres alternativas para dichas páginas web: "uam.es", "hpcn-uam.es", "20minutos.es".

En la Figura 4-7, se muestra la lista completa de dispositivos simulados dentro del fichero Emuladores.py:

```
"Apple iPhone 4",
"Apple iPhone 5",
"Apple iPhone 6",
"Apple iPhone 6 Plus",
"BlackBerry Z30",
"BlackBerry PlayBook",
"Google Nexus 4",
"Google Nexus 5",
"Google Nexus 6",
"Google Nexus 7",
"Google Nexus 10",
"LG Optimus L70",
"Nokia Lumia 520",
"Nokia N9",
"Samsung Galaxy Note 3",
"Samsung Galaxy Note II",
"Samsung Galaxy S4",
"Samsung Galaxy S III"
```

Figura 4-7. Lista de dispositivos simulados en el fichero Emuladores.py.

La aplicación mantiene un registro sobre todos los dispositivos que han sido detectados durante su ejecución. Para ello, se hace uso de una variable de estado dentro de la tabla hash encargada de almacenar los terminales identificados. De esta forma, es posible conocer de un modo directo si un dispositivo concreto se encuentra actualmente en la red, o bien si ha estado conectado, pero ya la ha abandonado. La lógica a seguir se aplica de la siguiente forma:

- Si el registro de la tabla se encuentra vacío, la variable de estado tiene un valor '0'.
- Si el terminal está conectado a la red actualmente, el valor es igual a '1'.
- Por último, si el dispositivo ya no se encuentra en la red, su estado es '2'.

Esta implementación permite tener localizados a aquellos dispositivos que han pasado por la red bajo estudio a lo largo del tiempo, haciendo posible el seguimiento sobre su ocupación y su actividad. Si, por ejemplo, un terminal abandonase la red y volviese a ella al cabo de un tiempo, con esta lógica sería posible conocer esta circunstancia y reflejarla en los registros de salida, ya que, si un dispositivo no es detectado en las tablas o su información se considera obsoleta, se inserta una nueva entrada en ellas.

El prototipo actual cuenta con una base de datos interna de terminales conocidos que se utiliza para identificar el modelo al que pertenecen una serie de dispositivos dados de un modo directo y exacto. Se trata de un conjunto de dispositivos móviles cuya firma ha sido identificada de cara a la realización de este Proyecto, facilitando así la labor realizada por la aplicación y otorgando a la misma un mayor grado de acierto frente a la llegada de nuevos dispositivos. El administrador de la red tiene la posibilidad de completar esta lista con nuevos elementos si lo considera conveniente, añadiendo nuevas entradas en las tablas en la parte del código de la aplicación encargada de la inserción de firmas a la base de datos.

En la Tabla 1 se puede ver el listado de firmas conocidas del que dispone la versión actual de la aplicación.

DISPOSITIVO	FIRMA ASOCIADA	FABRICANTE	SO
Galaxy Note 3	*SM-N900T*	*Samsung*	*Android*
Galaxy Note II	*GT-N7100*		
Galaxy S III	*GT-I9300*		
Galaxy S5	*SM-G900P*		
Samsung Galaxy S6	*SM-G920V*		
Samsung Galaxy S6 Edge Plus	*SM-G928X*		
LG Optimus L70	*LGMS323*	*LG*	
Nexus 4	*Nexus 4*	*Google Nexus*	
Nexus 5	*Nexus 5*		
Nexus 6	*Nexus 6*		
Nexus 6P	*Nexus 6P*		
Nexus 7	*Nexus 7*		
Nexus 10	*Nexus 10*		
BQ Aquaris X5	*Aquaris X5*	*BQ*	
Sony Xperia U	*SonyEricssonST25i*	*Sony*	
Sony Xperia Z5	*E6653*		
HTC One M9	*HTC One M9*	*HTC*	
HTC Desire Z	*HTC_DesireZ_A7272*		
Alcatel One Touch 4010D	*ALCATEL ONE TOUCH 4010D*	*Alcatel*	
Huawei Honor 3C	*H30-T00*	*Huawei*	
Motorola Moto G	*XT1032*	*Motorola*	
BlackBerry Z30	*BB10; Touch*	*BlackBerry*	*BlackBerry OS*
Nokia N9	*MeeGo; NokiaN9*	*Nokia*	*MeeGo*

Tabla 1. Listado de dispositivos reconocidos.

Como se puede apreciar, no existe mucha homogeneidad en las cadenas utilizadas entre los distintos fabricantes, incluso para un mismo Sistema Operativo. Mientras que algunas compañías, como *Nexus* o *Alcatel*, especifican el modelo de dispositivo literalmente en el User-Agent, otras como *Sony* o *Motorola* utilizan nombres poco intuitivos para sus terminales. Por estas razones, resulta de gran importancia realizar un estudio detallado y exhaustivo de los dispositivos más empleados a nivel global si se desea aprovechar el potencial de la aplicación.

El programa dispone de tres grandes módulos funcionales, dependiendo del Sistema Operativo. Los dos principales corresponden a los SOs Android e iOS, mientras que el restante se emplea para todos los demás dispositivos. En el listado de terminales reconocidos no se ha incluido ninguno con Sistema Operativo iOS, puesto que la identificación se realiza de forma directa a partir de su User-Agent; además, como el número de versiones diferentes de teléfonos móviles desarrollados por la compañía *Apple* hasta la fecha no es muy elevado, resulta asumible realizar la identificación directamente en el módulo correspondiente, sin tener que consultar la base de datos de la aplicación.

En las tablas se han incluido dos firmas correspondientes a dispositivos cuyo Sistema Operativo no es ninguno de los indicados anteriormente (*BlackBerry Z30* y *Nokia N9*). Para los dispositivos del módulo funcional "Otros", se analiza la cadena completa que hay dentro del paréntesis del UA. Como no existe homogeneidad entre los proveedores de SOs, se ha optado por tener en cuenta la cadena completa de cara a la identificación. La gestión de este tipo de terminales se realiza mediante tablas diferentes a las empleadas para dispositivos Android.

5 Desarrollo de la aplicación

5.1 Pasos seguidos para la implementación

Antes de comenzar con el desarrollo del código de la aplicación implementada, se realizó una profunda investigación sobre el caso de estudio del Trabajo. Este estudio sirvió para identificar las pautas a tener en cuenta y el camino a seguir para la implementación.

Inicialmente, se realizó una aproximación inicial utilizando distintos emuladores mediante los navegadores *Firefox* y *Chrome* para identificar patrones y características concretas de cada tipo de dispositivo. A partir de este punto, se comenzó con el desarrollo del código, que, dentro del Trabajo, fue la parte a la que se dedicaron más recursos.

En cuanto a la metodología empleada durante la implementación, los esfuerzos se centraron en cumplir los objetivos marcados a corto/medio plazo en cada una de las reuniones fijadas de forma periódica. Podríamos identificar las siguientes fases durante el desarrollo:

- En primer lugar, determinar las funcionalidades requeridas, las necesidades a cubrir y el tipo de interacción con los usuarios deseado en la aplicación.

- Identificar periódicamente los apartados de mayor relevancia para priorizar su implementación respecto al resto, estableciendo así los siguientes objetivos a alcanzar, y adaptar el prototipo en función de los resultados obtenidos y de las nuevas funcionalidades identificadas.

- Planificar el trabajo y llevar a cabo la implementación del código de cada una de las partes establecidas.

- Una vez efectuados los avances en la implementación, realizar las pruebas oportunas sobre los emuladores diseñados, con el fin de analizar su comportamiento y poder reparar así posibles fallos en el diseño.

- Revisar el funcionamiento en las reuniones presenciales y adecuarlo siguiendo las instrucciones dadas.

5.2 Estructura del proyecto

La aplicación está estructurada en 5 ficheros, codificados con el lenguaje de programación C, en los que se encuentra organizada la lógica de la misma. Los ficheros en los que está organizado el proyecto son los siguientes:

- HTTP_traffic_analysis.c

  Se trata del fichero principal de la aplicación. En él se gestiona el flujo de ejecución de la misma y se realizan las operaciones más relevantes relacionadas con el análisis de tráfico. También incluye algunos de los parámetros de configuración de la aplicación, como el tiempo tras el cual se imprime el contenido de las tablas de dispositivos reconocidos o la longitud máxima admisible de los paquetes capturados.

  Cuando se lanza el programa, se establece, en primer lugar, la configuración del mismo y se inicializan las tablas utilizadas para almacenar las firmas conocidas de la aplicación y los modelos encontrados durante su ejecución. A continuación, se comprueba el tipo de traza que se desea analizar y, en caso de tratarse de una traza *offline*, se abre el archivo con extensión .pcap indicado dentro del propio código y se comienza con el tratamiento individualizado de cada paquete. Si, por el contrario, se desea realizar las operaciones sobre el tráfico capturado en tiempo real, se abre la interfaz correspondiente y se aplican los filtros relacionados con el tráfico HTTP.

  El flujo seguido por cada llegada de un nuevo paquete al sistema es siempre el mismo. En primer lugar, se realizan algunas comprobaciones sobre los protocolos utilizados y la longitud de las cabeceras, con el fin de descartar aquellos paquetes que no son válidos o no tienen interés para la aplicación. El siguiente paso consiste en extraer información relevante sobre el paquete recibido (por ejemplo, las direcciones IP y MAC y los puertos de origen y destino o su longitud) y mostrarla por pantalla.

  Tras realizar estas comprobaciones, se comienza con el análisis del *payload* del paquete. En primer lugar, se eliminan los caracteres no legibles del mismo para facilitar el tratamiento de las cadenas de texto que se lleva a cabo en la aplicación. Posteriormente, se comprueba si el paquete HTTP contiene entre sus cabeceras el campo correspondiente al "User-Agent", puesto que, de no ser así, no es posible determinar de qué dispositivo se trata, por lo que se finaliza con el análisis de dicho paquete y el programa continúa analizando nuevas tramas.

  Más tarde, se busca entre las cabeceras el contenido del campo "Referer", que identifica la página web de origen de la petición. Una vez extraídos estos datos, se analiza el contenido del User-Agent en busca de alguno de los patrones conocidos para los dispositivos móviles incluidos en las bases de datos iniciales. Para ello, se realiza un procesamiento de cierto grado de complejidad de las cadenas de texto resultantes y, en función del Sistema Operativo detectado, se envía la ejecución al módulo adecuado en cada caso, teniendo en cuenta que los módulos incluidos en la versión actual de la aplicación corresponden a los SOs Android, iOS u otros.

  Dentro de cada módulo, se siguen las pautas establecidas en cada caso para ser capaces de identificar el dispositivo concreto al que corresponde el paquete y, con toda la información recaudada, se inserta el modelo detectado en las tablas correspondientes.

  Finalmente, se comprueba si ha transcurrido el tiempo suficiente desde la última impresión del contenido de las tablas con los dispositivos reconocidos y, en caso

afirmativo, se realiza una nueva impresión de las mismas en los ficheros de salida y en pantalla.

- HTTP_traffic_analysis.h

  Es el archivo de cabecera o *header file* en el que se encuentra la declaración de las clases, estructuras y variables utilizadas en el fichero *HTTP_traffic_analysis.c*. Contiene la declaración de todas las funciones empleadas por dicho archivo y las definiciones de las cabeceras IP y TCP.

- tablasDispositivos.c

  Es el fichero encargado de implementar las funciones relacionadas con la identificación de nuevos dispositivos en la red.

  Entre sus métodos, se incluye el correspondiente a la inserción de un nuevo elemento en las tablas. En él, se comprueba si dicho elemento ya se encuentra en las bases de datos, puesto que, de ser así, únicamente se actualizan los valores correspondientes a la hora de llegada del último paquete, su identificador, la web a visitar y el total de bytes recibidos durante la conexión de dicho dispositivo. Si, por el contrario, éste acaba de conectarse a la red, se genera una nueva entrada en la tabla y se almacenan todos los parámetros relacionados con la visita.

  También incluye la función encargada de imprimir el contenido de las tablas. Esta función vuelca simultáneamente el contenido las tablas de dispositivos en dos ficheros, uno con extensión .csv y otro con extensión .log, haciendo accesible la información recopilada sobre los datos específicos de la red para los usuarios. Al mismo tiempo, se muestra en la consola de ejecución un listado de los paquetes recibidos, de modo que se dispone de un registro muy completo de la actividad desarrollada.

  Asimismo, se incluyen las funciones utilizadas para crear y vaciar las tablas, así como para determinar la posición de la tabla en que debe ser almacenado un nuevo elemento.

- tablasFirmas.c

  Incluye la lógica de las funciones encargadas de la gestión de las firmas conocidas de la aplicación. Contiene los métodos utilizados para crear, inicializar y buscar en patrones conocidos las tablas para los módulos funcionales de los SO Android y "Otros", así como los métodos empleados para generar la posición de la tabla en la que se va a almacenar o buscar un registro.

- tablas.h

  En él se encuentran las cabeceras correspondientes a la gestión de las tablas utilizadas a modo de base de datos de la aplicación, incluyendo las relacionadas con las firmas conocidas y con los dispositivos identificados.

  En este fichero se definen las estructuras en las que se almacenarán los dispositivos de la base de datos inicial de la aplicación y la información generada a medida que se reconozcan nuevos dispositivos, así como la longitud reservada para cada entrada de las tablas. Además, también recoge la declaración de las funciones incluidas en los ficheros *tablasFirmas.c* y *tablasDispositivos.c.*

Adicionalmente, para la implementación de la aplicación, se emplearon dos ficheros para realizar las distintas pruebas llevadas a cabo durante la fase de desarrollo.

- Emuladores.py

  Este fichero genera visitas a una serie de páginas web utilizando para ello un conjunto de emuladores predefinidos, que simulan el funcionamiento de dispositivos reales y utilizan un navegador web para el acceso a Internet. En este caso, se hace uso de las opciones para desarrolladores que incluye el navegador Chrome.

  Cuando se lanza el ejecutable, se configuran, en primer lugar, los parámetros correspondientes a los terminales móviles a simular. El programa dispone de una lista de dispositivos que se va recorriendo durante la ejecución, generando una visita a una página web para cada uno de ellos. Tras llevar a cabo la configuración inicial, se comienza a recorrer la lista de dispositivos y, por cada uno, se establece de forma aleatoria la web a visitar y el tiempo de visita. Una vez transcurrido dicho periodo de tiempo, se cierra la pestaña del navegador y se avanza al siguiente dispositivo.

  Los datos sobre cada una de las visitas realizadas se muestran por pantalla y, al mismo tiempo, se vuelca en un fichero de salida. De esta forma, se consigue mantener un registro de la actividad que ha tenido lugar a modo de *Ground Truth* para poder consultar los datos reales y evaluar la precisión de la aplicación una vez finalizada su ejecución.

- ejecutar.sh

  Se trata de un script *bash* cuya única función es lanzar indefinidamente el contenido del fichero Emuladores.py, generando así un conjunto ilimitado de visitas a páginas web que podrá ser utilizado para realizar las pruebas oportunas.

Todo el código desarrollado para este Trabajo se encuentra disponible en el siguiente repositorio de *GitHub*: *https://github.com/ablazquez/TFM_ABlazquez-Identify-Mobiles*. Se ha optado por dar a la aplicación el carácter de *open source*, de manera que se pueda contribuir libremente a su desarrollo, aportar mejoras o proponer nuevas funcionalidades a

la misma. En caso de cualquier duda concreta sobre cuestiones específicas relacionadas con la programación de la aplicación, se recomienda acudir al repositorio indicado y comprobar directamente en él el código implementado.

5.3 Requisitos de la aplicación

Para poder utilizar la aplicación desarrollada en un router doméstico, son varias las consideraciones que hay que tener en cuenta para que la ejecución sea correcta. A continuación, se exponen algunos de los puntos más importantes a considerar.

En primer lugar, debe existir una carpeta denominada /*registros* dentro de la ruta base de la aplicación. En dicha ubicación es donde se almacenan los ficheros generados durante la ejecución. Se ha optado por la utilización de esta carpeta para almacenar los registros generados por el programa con el fin de tener organizados los ficheros generados y poder garantizar su accesibilidad de cara a los usuarios finales del mismo.

Es recomendable que los resultados de la ejecución del programa se vuelquen en un fichero de texto. De esta forma, es posible mantener un registro de los datos originados por la aplicación durante su ejecución para analizarlos una vez finalizada la misma. Si no se tuviese en cuenta esta indicación, habría cierta información que aparecería por pantalla en lugar de quedar almacenada, lo cual puede ser igualmente de utilidad, aunque se recomienda optar por esta última opción. La instrucción que se propone para ejecutar el programa es la siguiente:

sudo ./HTTP_traffic_analysis > registros/TRAFICO.log

Por otra parte, si se realizan cambios en el código de la aplicación, se necesita un compilador cruzado para que el ejecutable generado sea compatible con la arquitectura del posible router sobre el que será lanzada. Esta cuestión es necesaria debido a que la plataforma sobre la que se ejecuta el programa no es la misma que la utilizada para compilarlo. A través de un compilador cruzado, se consigue que el router sea capaz de lanzar el archivo generado y que éste incorpore, por ejemplo, las librerías de libpcap, que son necesarias para la utilización de la aplicación. El compilador utilizado para este Trabajo ha sido el proporcionado por *Buildroot* a través de su página oficial. En el caso del router empleado para las pruebas realizadas sobre la aplicación, la arquitectura sobre la que estaba basada su CPU era de tipo MIPS. Por tanto, el compilador cruzado empleado fue configurado para trabajar sobre esta familia de microprocesadores.

Por último, es conveniente destacar que se necesitan permisos de administrador para poder lanzar el programa. Esto es debido a que las librerías de *libpcap* realizan una búsqueda de interfaces sobre las que trabajar y, sin los permisos adecuados, esta funcionalidad queda bloqueada, impidiendo la ejecución normal de la aplicación. No obstante, al entrar a la configuración del router para lanzar el programa, el acceso se realiza siempre con dichos permisos, por lo que, en principio, no es necesario tener en cuenta esta indicación si el programa se ejecuta sobre el propio router.

6 Validación

6.1 Proceso de instalación

Una vez completado todo el desarrollo de la aplicación, el último paso previsto dentro del Proyecto fue su puesta en marcha sobre el router encargado de gestionar una red concreta, con el fin de validar su funcionamiento en un entorno real. Para ello, se realizaron pruebas sobre el router ***Cisco Linksys E3000*** (Figura 6-1), disponible dentro de la Escuela Politécnica Superior. Se trata de un router comercial con coste de alrededor de 100€, fabricado por la compañía *Cisco*, que ofrece un total de 4 puertos LAN y uno WAN, soportando velocidades en cada uno de ellos de 1,000 Mb/s. La instalación se realizó teniendo en cuenta las características particulares del enrutador a utilizar, empleando el software que se indica en este Apartado.

Figura 6-1: Router Cisco Linksys E3000.

En el modelo disponible en la EPS se remplazó el sistema operativo nativo por otro, aunque también basado en un sistema Linux, permitiendo total libertad para instalar y ejecutar aplicaciones. En concreto, se realizó la instalación del firmware ***OpenWrt*** [29], que está basado en una distribución de Linux empotrada y permite la instalación de SOs en dispositivos como routers domésticos.

El instalador de paquetes empleado fue ***OPKG***, que es un gestor de paquetes ligero que se utiliza para descargar e instalar paquetes de *OpenWrt* desde repositorios locales o situados en Internet.

La memoria del router fue gestionada a través de una unidad externa USB, empleada como una extensión al almacenamiento. Para ello, se utilizó ***OverlayFS***, una implementación de sistema de archivos de montaje de unión para Linux que permite montar el dispositivo USB como una unidad básica, utilizada a modo de disco duro.

Por su parte, en la máquina local se instaló un compilador cruzado para la arquitectura de la CPU del router (MIPS): ***Buildroot***. Este tipo de compilador es necesario debido a que la arquitectura del router, que es donde se lanzará el programa, no es la misma que la

utilizada para su compilación. Lo que hace *Buildroot* es preparar el proyecto para ser ejecutado en dicha plataforma y generar el ejecutable de acuerdo a las restricciones de la arquitectura objetivo.

El acceso al router en remoto se realizó mediante conexiones de tipo ***ssh***. A través del terminal de la máquina virtual Linux, la instrucción ssh permitió acceder a un servidor privado a través de un *backend* utilizando el protocolo de mismo nombre, SSH. El router fue configurado dentro de la ruta *@argon.ii.uam.es* para ser accesible desde cualquier punto en el exterior.

Una vez iniciada la sesión contra el router, el ejecutable de la aplicación fue copiado en él mediante el comando ***scp***. De igual modo, una vez terminada la ejecución del programa en el router, la recuperación de los registros generados para su posterior análisis se realizó también a través de este comando. La utilización de scp hizo posible la transferencia segura de ficheros en ambas direcciones entre el host remoto y la máquina local, basándose para ello en el protocolo SSH.

Teniendo en cuenta todas las consideraciones mencionadas, se llevó a cabo el proceso completo de instalación y, una vez finalizado, se realizaron distintas pruebas sobre la aplicación desarrollada conectando simultáneamente varios terminales móviles a la red Wifi levantada por el router.

Idealmente, el software desarrollado debería ser instalado por defecto en los routers domésticos entregados a los usuarios por parte de los operadores, de forma que ya tuviesen integrada la aplicación desde el primer momento. De este modo, los usuarios no tendrían la necesidad de realizar ninguna operación sobre ellos y serían los propios operadores quienes se encargarían de recoger los registros generados por la aplicación. Por tanto, la información sería accesible únicamente para los operadores y éstos tendrían la posibilidad de tomar decisiones en base a la actividad que se detecte por parte de los usuarios mediante la aplicación. En cualquier caso, este proceso de modificación es muy común en los operadores, que entregan routers listos para ser utilizados únicamente en sus redes o configurados para separar datos y voz en clientes que contratan ambos servicios.

6.2 Resultados obtenidos

La validación final de la aplicación se realizó tras su ejecución como un sistema final listo para su entrada en operación comercial. La versión definitiva del prototipo desarrollado fue puesta en marcha sobre un router a modo de prototipo con el fin de analizar su comportamiento y validar su funcionamiento sobre un escenario real, conectando para ello varios dispositivos a la red y navegando por Internet con ellos para generar el tráfico web correspondiente.

Para las pruebas finales llevadas a cabo, se lanzó la aplicación mientras se encontraban conectados a la red Wifi dos terminales móviles de uso personal: un *BQ Aquaris X5* y un *Motorola Moto X Play XT1532*. A partir del tráfico que generaron ambos dispositivos en la red, se pudo generar un escenario similar al que podría darse en una red real.

Una de las cuestiones que permitió descubrir la instalación de la aplicación en el router es que la interfaz de captura debe ser aquella a la que se conectan directamente los dispositivos. En el caso del router empleado, dicha interfaz corresponde a *wlan0*. Los paquetes generados por los dispositivos móviles atraviesan también otras interfaces, como *eth0.1*, *eth0.2* o *brlan*, y en todas ellas son visibles, pero no mantienen todas sus propiedades. Por ejemplo, estas interfaces añaden a sus paquetes cabeceras correspondientes a la capa VLAN, definida en la capa de enlace de datos (nivel 2 del modelo OSI), agregando en la cabecera de la trama Ethernet 4 bits que conforman el identificador de VLAN (Figura 6-2). Cuando esto ocurre, el filtrado de paquetes deja de aplicarse del mismo modo y, además, las direcciones IP y MAC de los paquetes dejan de ser las de los dispositivos que los generaron, pasando a ser las del propio router, que será el encargado de transmitirlos recibir su respuesta, por lo que se pierde la referencia de quién los originó en realidad.

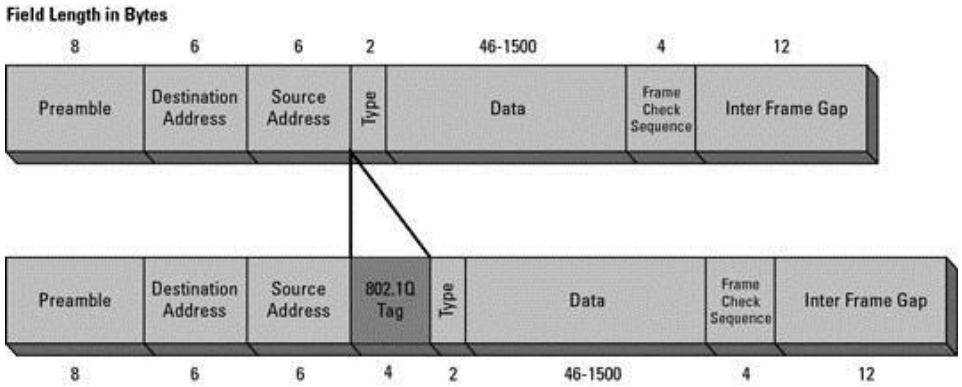

Figura 6-2. Cabeceras VLAN en trama Ethernet.

(Fuente: *https://es.wikibooks.org/wiki/Planificación_y_Administración_de_Redes/Tema_9/Conceptos_generales*)

Con todas estas consideraciones, se lanzó la aplicación de manera remota, utilizando el protocolo ssh para realizar la conexión entre el equipo y el router. Durante el tiempo de ejecución, se realizaron visitas a diferentes páginas web desde los teléfonos móviles conectados a la red Wifi para generar tráfico en ella. Al mismo tiempo, se realizó una captura de tráfico sobre los paquetes generados. Dicha traza de captura, que puede resultar de utilidad útil para futuras pruebas, se incluye en la entrega de este Trabajo bajo el nombre de *eth0.1499102166.wlan0.pcap*.

Los resultados obtenidos fueron muy satisfactorios, puesto que ambos dispositivos fueron reconocidos e identificados de manera correcta. Pese a que el entorno de pruebas era reducido, los datos obtenidos dan una muestra de la fiabilidad de la aplicación y permiten ver de qué modo funciona y cuáles son los registros que genera durante una ejecución normal.

En la Figura 6-3, Figura 6-4 y Figura 6-5 se puede observar una parte del contenido de los siguientes ficheros tras la ejecución en el router:

- TRAFICO.log.
- HISTORICO_DISPOSITIVOS.log.
- DISPOSITIVOS_RECONOCIDOS.csv.

Tras analizar todas las pruebas realizadas, se puede afirmar que la aplicación funciona de un modo muy adecuado y posee un potencial notablemente elevado de cara a una posible utilización en routers comerciales, por su previsible fiabilidad y por la información de valor que aporta a los administradores de red. Se puede concluir, por tanto, que el prototipo actual se encuentra listo para ser desplegado y cumple con todos los objetivos marcados dentro del Proyecto.

```
2506
2507 Paquete número: 203
2508 IP origen: 192.168.1.220
2509 IP destino: 150.244.214.237
2510 MAC origen: 4c:74:3:f4:8b:30
2511 MAC destino: c0:c1:c0:4c:af:9e
2512 Puerto origen: 49741
2513 Puerto destino: 80
2514 Longitud: 819
2515 Hora: Mon Jul  3 19:16:17 2017
2516 Timestamp: 1499102177
2517 User-Agent: Mozilla/5.0 (Linux; Android 6.0.1; Aquaris X5 Build/MMB29M)
     AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Mobile Safari/537.36
2518
2519 ***MODELO DETECTADO: BQ Aquaris X5***
2520
2521
2522
2523 Paquete número: 204
2524 IP origen: 150.244.214.237
2525 IP destino: 192.168.1.220
2526 MAC origen: c0:c1:c0:4c:af:9e
2527 MAC destino: 4c:74:3:f4:8b:30
2528 Puerto origen: 80
2529 Puerto destino: 49741
2530 Longitud: 724
2531 Hora: Mon Jul  3 19:16:17 2017
2532 Timestamp: 1499102177
2533
2534
2535 Paquete número: 205
2536 IP origen: 150.244.214.237
2537 IP destino: 192.168.1.220
2538 MAC origen: c0:c1:c0:4c:af:9e
2539 MAC destino: 4c:74:3:f4:8b:30
2540 Puerto origen: 80
2541 Puerto destino: 49741
```

Figura 6-3. TRAFICO.log: Resultados tras la ejecución en el router.

```
        -- DISPOSITIVOS RECONOCIDOS --
        Mon Jul  3 19:11:20 2017

1. MODELO: Motorola Moto X Play XT1562
Web a visitar: http://www.elmundo.es/
MAC: 84:10:d:99:45:b6
Dirección IP: 192.168.1.114
Timestamp 1er paquete: 1499102174
Hora 1er paquete: Mon Jul  3 19:10:14 2017
Timestamp último paquete: 1499102176
Hora último paquete: Mon Jul  3 19:11:18 2017
Total bytes intercambiados: 11313
Modelo conocido



        -- DISPOSITIVOS RECONOCIDOS --
        Mon Jul  3 19:16:20 2017

1. MODELO: Motorola Moto X Play XT1562
Web a visitar: http://www.elmundo.es/
MAC: 84:10:d:99:45:b6
Dirección IP: 192.168.1.114
Timestamp 1er paquete: 1499102174
Hora 1er paquete: Mon Jul  3 19:10:14 2017
Timestamp último paquete: 1499102180
Hora último paquete: Mon Jul  3 19:16:20 2017
Total bytes intercambiados: 90117
Modelo conocido

2. MODELO: BQ Aquaris X5
Web a visitar: http://www.uam.es/UAM/Home.htm?language=es
MAC: 4c:74:3:f4:8b:30
Dirección IP: 192.168.1.220
Timestamp 1er paquete: 1499102177
Hora 1er paquete: Mon Jul  3 19:16:17 2017
Timestamp último paquete: 1499102178
Hora último paquete: Mon Jul  3 19:16:18 2017
Total bytes intercambiados: 27323
Modelo conocido
```

Figura 6-4. HISTORICO_DISPOSITIVOS.log: Resultados tras la ejecución en el router.

MODELO	ACTIVO	WEB A VISITAR	DIRECCION IP	MAC	TIMESTAMP 1er PAQUETE	HORA 1er PAQUETE	TIMESTAMP ULTIMO PAQUETE	HORA ULTIMO PAQUETE	TOTAL BYTES INTERCAMBIADOS	MODELO CONOCIDO
Motorola Moto X Play XT1562	SI	http://www.elmundo.es/	192.168.1.114	84:10:d:99:45:b6	1499102174	Mon Jul 3 19:16:14 2017	1499102180	Mon Jul 3 19:16:20 2017	30117	SI
BQ Aquaris X5	SI	http://www.uam.es/UAM/Home.htm	192.168.1.220	4c:74:3:f4:8b:30	1499102177	Mon Jul 3 19:16:17 2017	1499102178	Mon Jul 3 19:16:18 2017	27323	SI

Figura 6-5. DISPOSITIVOS_RECONOCIDOS.csv: Resultados tras la ejecución en el router.

7 Conclusiones y líneas futuras

7.1 Conclusiones

La evolución tecnológica actual obliga a los operadores móviles a utilizar nuevos mecanismos para que los servicios que prestan a sus usuarios sean eficientes y se ajusten a los parámetros exigibles según los SLAs establecidos con sus clientes en un mercado cada vez más competitivo. Esto les obliga a dedicar muchos esfuerzos en mantener un control completo sobre sus redes, empleando para ello nuevas técnicas y tratando de aprovechar eficientemente todos los recursos de los que disponen.

El objetivo principal del Proyecto era determinar si es posible identificar qué tipo de dispositivos emplean los usuarios de una red Wifi y, una vez establecido el camino a seguir, implementar las herramientas precisas y realizar una prueba de concepto sobre ellas. Todas estas metas marcadas a priori han sido cumplidas de manera satisfactoria, alcanzando resultados precisos y convincentes, por lo que podemos calificar la consecución de las mismas como exitosa.

Se partía de una incertidumbre inicial bastante elevada sobre si sería factible identificar con precisión los modelos de los terminales móviles, puesto que no se arrancaba a partir de ningún trabajo previo y no existía ninguna referencia clara sobre qué pautas seguir para alcanzar los objetivos. El trabajo de investigación previo a la implementación de la aplicación permitió, en primer lugar, conocer con mayor profundidad el funcionamiento interno de las redes de telefonía móvil y, por otro lado, entender cómo se produce el intercambio de tráfico dentro de una red Wifi. A partir de este conocimiento, se pudieron determinar las reglas que se debían aplicar para la identificación y trasladarlas al código fuente del programa.

El prototipo desarrollado se podría definir como un producto final, listo para ser desplegado. El atractivo de este producto es que cualquier operador podría estar interesado en desplegarlo en sus routers comerciales, teniendo así acceso a un mayor conocimiento sobre la actividad llevada a cabo por los elementos que se encuentran bajo su dominio, lo cual era el objetivo final del Proyecto. Por tanto, el *software* cumple con los requisitos establecidos y proporciona una información muy valiosa a sus usuarios.

El funcionamiento ha sido validado sobre un router en un entorno controlado, dando lugar a unos resultados muy satisfactorios. La aplicación no ha llegado a ser probada en términos de rendimiento, puesto que, para ello, debe ser ejecutada en los equipos finales de los usuarios sin mayor agregación. No obstante, se han añadido muchas funcionalidades adicionales que se han considerado interesantes de cara a los administradores de red y que la hacen realmente atractiva desde el punto de vista comercial, aunque pueden ser descartadas en caso de que el *hardware* muestre cierta sobrecarga.

A nivel personal, este Proyecto me ha servido para adquirir algunos conceptos y reforzar muchos otros, así como para dominar, en general, metodologías que desconocía. En concreto, podría destacar el aprendizaje sobre el lenguaje de programación Python, las librerías de captura de tráfico proporcionadas por libpcap, la utilización de tablas hash, el protocolo HTTP, la implementación de software para hardware concreto de baja capacidad

(como lo es un router doméstico), etc. Considero que, en ese sentido, ha sido un trabajo muy enriquecedor, puesto que dichos conocimientos podrán resultarme de gran utilidad en el futuro.

En general, el trabajo realizado ha sido útil para ampliar mi capacidad de tomar decisiones y afrontar y resolver los problemas surgidos. Por todo ello, puedo afirmar que me encuentro satisfecho con la labor que he llevado a cabo, teniendo en cuenta los resultados alcanzados y la forma en que éstos han sido logrados.

7.2 Líneas futuras

La versión actual de la aplicación cubre las necesidades básicas que se plantearon inicialmente e incluye diversas funcionalidades que se han considerado interesantes desde el punto de vista práctico para su utilización por parte de los gestores de red. No obstante, aún existe margen de mejora sobre aspectos de funcionamiento que se podrían revisar o posibles nuevas características para mejorar la experiencia de los usuarios.

Por ejemplo, algunas de las líneas futuras de trabajo que se proponen son las siguientes:

- Implementar un *script* que compruebe el tamaño de los ficheros generados en el router para garantizar que no van a surgir problemas de memoria.

- Validar el funcionamiento de la aplicación sobre una red en la que se encuentre conectado un número considerable de usuarios y realizar un análisis de los resultados obtenidos.

- Limpiar el contenido de las tablas de dispositivos periódicamente (por ejemplo, al finalizar el día).

- Añadir nuevos registros a las tablas de dispositivos reconocidos para aumentar la exactitud de los resultados finales. El prototipo actual ha sido desarrollado con en torno a 30 dispositivos conocidos para realizar las pruebas oportunas, pero si se optase por dar un enfoque comercial al mismo, sería interesante contar con un mayor número de firmas conocidas sobre las que comparar.

- Identificar patrones que indiquen si alguno de los dispositivos identificados es un equipo conocido como “no móvil”. Por ejemplo, se podría tener en consideración algún tipo de pauta que identificase con certeza la presencia de ordenadores con Sistema Operativo Linux, Windows, etc.

- Comprobar el contenido de los registros con extensión .csv y controlar la existencia de caracteres como ‘;’ o ‘,’, sustituyéndolos por cualquier otro carácter a convenir, o bien simplemente eliminándolos, puesto que pueden dar lugar a separaciones erróneas entre los distintos campos.

- Solicitar al usuario final algunos de los parámetros de configuración con cada ejecución (periodicidad del tiempo tras el que se imprimen las tablas, tamaño de las mismas, etc.).

- Realizar una representación gráfica de la evolución de la red de manera automática que ayude a comprender la utilización de la misma en función del tiempo.

Referencias

[1] M. Lee. “Google ads and the blindspot debate”, Media, Culture & Society 33(3): 433-447, 2011.

[2] M. Sullivan. “What Your Wireless Carrier Knows About You”, PCWorld, 2011. Available: http://www.pcworld.com/article/228813/what_your_wireless_carrier_knows_about_you.html

[3] GSM Association, “The Mobile Economy”, GSMA: 6-9, 2017. Available: https://www.gsmaintelligence.com/research/?file=9e927fd6896724e7b26f33f61db5b9d5&download

[4] GSM Association, “Informe Mobile en España y en el Mundo 2017”, GSMA, 2017.

[5] Computer Crime and Intellectual Property Section, U.S. Department of Justice, “Retention Periods of Major Cellular Services Providers”, August, 2010. Retrieved from: https://www.aclu.org/files/pdfs/freespeech/retention_periods_of_major_cellular_service_providers.pdf

[6] IEEE Computer Society, “802.11ai-2016 —Telecommunications and information exchange between systems - Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 1: Fast Initial Link Setup”. Available: http://standards.ieee.org/getieee802/download/802.11-2016.pdf

[7] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, “Hypertext Transfer Protocol -- HTTP/1.1”, IETF, June, 1999. Available: http://www.rfc-base.org/rfc-2616.html

[8] D. Manners, “The User Agent Field: Analyzing and Detecting the Abnormal or Malicious in your Organization”, The Sans Institute: 6-12, 2012. Available: https://www.sans.org/reading-room/whitepapers/malicious/user-agent-field-analyzing-detecting-abnormal-malicious-organization-33874

[9] Wireshark Foundation, “OUI Lookup Tool”. Available: https://www.wireshark.org/tools/oui-lookup.html

[10] M. Finsterbusch, C. Richter, E. Rocha, J. Muller, and K. Hanbge. “A Survey of Payload-Based Traffic Classification Approaches”. IEEE Communications Surveys & Tutorials, 16(2): 1135-1156, 2014.

[11] A. Boukhtouta, S. A. Mokhov, N. Lakhdari, M. Debbabi and P. Joey "Network malware classification comparison using DPI and flow packet headers", Journal of Computer Virology and Hacking Techniques 12(2): 69-100, 2016.

[12] nTop, “nDPI - Open and Extensible LGPLv3 Deep Packet Inspection Library”. Available: www.ntop.org/products/deep-packet-inspection/ndpi/

[13] T. Porter, “The Perils of Deep Packet Inspection”, SecurityFocus, 2005. Available: https://www.symantec.com/connect/articles/perils-deep-packet-inspection

[14] Google Developers Site, “User Agent Strings”. Available: https://developer.chrome.com/multidevice/user-agent

[15] R. Fielding, J. Reschke (eds.), “Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content”, IETF, June, 2014. Available: https://tools.ietf.org/html/rfc7231

[16] J. L. García-Dorado, P. M. Santiago del Río, J. Ramos, D. Muelas, V. Moreno, J. E. López de Vergara, J. Aracil. “Low-cost and High-performance: VoIP monitoring and

full-data retention at multi-Gb/s rates using commodity hardware", International Journal of Network Management 24(3):181-199, 2014.
[17] S. Miskovic, G. Moo Lee, Y. Liao, M. Baldi, "AppPrint: Automatic Fingerprinting of Mobile Applications in Network Traffic", Computer Science (8995), 2015. Available: http://wan.poly.edu/pam2015/papers/3.pdf
[18] Libelium Comunicaciones Distribuidas, "Smartphone, Cellular and Hands-Free Mobile Phone Detection". Available: http://www.libelium.com/products/meshlium/smartphone-detection/
[19] 51Degrees, "Device Detection". Available: https://51degrees.com/device-detection
[20] Andy Moore, "Detect Mobile Browsers - detect and redirect mobile browsers on your website". Available: https://detectmobilebrowsers.mobi/
[21] NETMARKETSHARE, "Mobile/Tablet Operating System Market Share". Available: https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1
[22] SeleniumHQ, "Selenium - Web Browser Automation". Available: http://www.seleniumhq.org
[23] Tcpdump/Libpcap, "TCPDUMP/LIBPCAP public repository". Available: http://www.tcpdump.org/
[24] Buildroot, "Buildroot - Making Embedded Linux Easy". Available: https://buildroot.org/
[25] Tao software, "tPacketCapture Application". Available: http://www.taosoftware.co.jp/en/android/packetcapture/
[26] Android Developers, "Build | Android Developers". Available: https://developer.android.com/reference/android/os/Build.html
[27] A. López, "Aprendiendo a programar con Libpcap", 2005. Available: http://e-ghost.deusto.es/docs/2005/conferencias/pcap.pdf
[28] Google Developers Site, "Chrome DevTools Overview". Available: https://developer.chrome.com/devtools
[29] OpenWrt, "Open Wrt – Wireless Freedom". Available: www.openwrt.org

Glosario

AP	Access Point
CPU	Central Processing Unit
DPI	Deep Packet Inspection
GSM	Global System for Mobile communications
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
LAN	Local Area Network
MAC	Media Access Control
MIPS	Microprocessor without Interlocked Pipeline Stages
OS	Operating System
OSI	Open System Interconnection
OUI	Organizationally Unique Identifier
RFC	Request For Comments
SCP	Secure Copy Protocol
SLA	Service Level Agreement
SSH	Secure SHell
UA	User Agent
USB	Universal Serial Bus
VLAN	Virtual Local Area Network
VM	Virtual Machine
WAN	Wide Area Network

Anexos

A Cabeceras HTTP

NOMBRE DE LA CABECERA	*DESCRIPCIÓN*	*EJEMPLO*	*ESTADO*
Accept	Content-Types (tipos de contenido) que se aceptan.	Accept: text/plain	Permanente
Accept-Charset	Conjunto de caracteres que se aceptan.	Accept-Charset: utf-8	Permanente
Accept-Encoding	Lista de codificaciones que se aceptan.	Accept-Encoding: gzip, deflate	Permanente
Accept-Language	Idiomas que se aceptan.	Accept-Language: en-US	Permanente
Accept-Datetime	Versión de la hora y fecha que se aceptan.	Accept-Datetime: Thu, 31 May 2007 20:35:00 GMT	Provisional
Authorization	Credenciales de autorización.	Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==	Permanente
Cache-Control	Se controla las políticas de caché.	Cache-Control: no-cache	Permanente
Connection	Se controla el tipo de conexión.	Connection: keep-alive	Permanente
		Connection: Upgrade	
Cookie	Una cookie enviada previamente por el servidor usando Set-Cookie	Cookie: $Version=1; Skin=new;	Permanente: Estándar
Content-Length	El tamaño del contenido de la petición en bytes	Content-Length: 348	Permanente
Content-MD5	Un checksum en MD5 sobre el contenido	Content-MD5: Q2hlY2sgSW50ZWdyaXR5IQ==	Obsoleto

Content-Type	El tipo de contenido de la petición en POST o PUT	Content-Type: application/x-www-form-urlencoded	Permanente
Date	La fecha y la hora de la petición	Date: Tue, 15 Nov 1994 08:12:31 GMT	Permanente
Forwarded	Indica la información original del cliente en caso de conexión por proxy.	Forwarded: for=192.0.2.60; proto=http; by=203.0.113.43 Forwarded: for=192.0.2.43, for=198.51.100.17	Permanente
From	La dirección de correo electrónico de la petición.	From: user@example.com	Permanente
Host	El nombre de dominio o dirección IP (puede incluir número de puerto). El uso de la cabecera es obligatorio a partir de HTTP 1.1	Host: en.wikipedia.org:8080	Permanente
Max-Forwards	Limita el número de veces que un mensaje viaja a través de los proxies.	Host: en.wikipedia.org Max-Forwards: 10	Permanente
Origin	Inicia una petición para servidores con respuesta a Access-Control-Allow-Origin.	Origin: http://www.example-social-network.com	Permanente: Estándar
Pragma	Implementa cabeceras en donde multiples efectos se aplica a todo.	Pragma: no-cache	Permanente
Proxy-Authorization	Credenciales de autorización para conectarse a un proxy.	Proxy-Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==	Permanente
Range	Pide sólo una parte del contenido	Range: bytes=500-999	Permanente
Referer [sic]	Indica la dirección URL de donde proviene, en otras palabras, es la dirección web del botón Atrás.	Referer: http://en.wikipedia.org/wiki/Main_Page	Permanente
User-Agent	Contiene la información de la petición, como el navegador, el sistema operativo, etc.	User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:12.0) Gecko/20100101 Firefox/21.0	Permanente

Upgrade	Pide al servidor que se actualice la versión de HTTP para funcionar.	Upgrade: HTTP/2.0, HTTPS/1.3, IRC/6.9, RTA/x11, websocket	Permanente
Warning	Una advertencia general sobre problemas de la entidad.	Warning: 199 Miscellaneous warning	Permanente

Tabla 2. Cabeceras de petición HTTP.

(Fuente: *https://es.wikipedia.org/wiki/Anexo:Lista_de_cabeceras_HTTP*)